NOVITÀ WEB
Finalmente on-line il sito ufficiale di ioProgrammo

SPECIALE HACKING
BUFFER OVERFLOW E ANALISI DEL SISTEMA

VERSIONE PLUS RIVISTA+LIBRO+CD €14,90
VERSIONE STANDARD RIVISTA+CD €6,90
Periodicità mensile • SETTEMBRE 2003 • ANNO VII, N.8 (72)
Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL

IO PROGRAMMO

ROBOT FAI DA TE!

Dalla realizzazione passo passo di un braccio meccanico alla sua programmazione

SISTEMA
- Un client di posta elettronica in Visual Basic
- FireFly: come connettere Flash e database senza scrivere codice
- Implementare Stored Procedure con ADO.NET

PALMARI
- Consultazione remota di dati aziendali
- La sincronizzazione dei dati con SQL Server CE

ADVANCED
- L'accesso ai WebServices con Java
- Java: uno strumento per valutare la qualità del nostro codice

CORSI
- Visual Basic .NET, JAVA, C++, C#, MATLAB

Previsioni meteo animate
Facile! Con Flash MX e Web Services

Microsoft .NET
Progettiamo un'applicazione in grado di autoaggiornarsi

Satellite e Palmari
GPS: il tuo Pocket PC come un navigatore satellitare

CROBOTS 2003: SCOPRI COME PARTECIPARE ALLA SFIDA TRA ROBOT CHE BATTAGLIANO A COLPI DI C++

ISSN 1128-594X
EDIZIONI MASTER
www.edmaster.it

ioProgrammo (plus) Anno VII - N° 8 (72) • € 14,90

Contents

Anno VII - n. 8 (72) Settembre 2003

www.ioprogrammo.net

Questo mese si risolve una grave anomalia… no, non stiamo parlando del conflitto di interessi, ma del fatto che ioProgrammo ha finalmente il suo sito web!
ioProgrammo è da molti anni la rivista di programmazione più venduta in Italia, ovviamente la speranza (convinzione?) è che il sito avrà lo stesso successo della rivista. La "missione" sarà quella che ci ha contraddistinto in tutti questi anni: essere uno strumento per chi programma "davvero".
Rigore e praticità sono i punti cardine della nostra idea di sviluppo, e tali rimarranno anche per il sito. Lì troverete articoli e contributi originali, anteprime e soprattutto un forum che sarà, ci auguriamo, il cuore pulsante del sito: un luogo dove scambiarsi opinioni e conoscenze, e dove gli autori di ioProgrammo saranno pronti a dialogare con voi e ad aiutarvi in caso di difficoltà.

Il sito ha avuto una lunga gestazione ma riteniamo che, anche grazie al lavoro dell'ottimo Piero Mannelli, tutto il tempo che gli è stato dedicato sia stato ben speso. Come sempre, per domande, suggerimenti e critiche siamo sempre a vostra disposizione.

P.S.
Scrivo queste note dal Tech Ed di Barcellona, edizione del decennale. Migliaia di sviluppatori e decine di sessioni tecniche di cui troverete puntuale cronaca nel prossimo numero di ioProgrammo. Vi anticipo i temi: Web Services, sicurezza, sicurezza… e ancora sicurezza!

raffaele@edmaster.it

ioPROGRAMMO

Anno VII - N.ro 8 (72) - Settembre 2003 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo

Direttore Editoriale Massimo Sesti
Direttore Responsabile Romina Sesti
Responsabile Marketing Antonio Meduri
Responsabile Editoriale Gianmarco Bruni
Editor Gianfranco Forlino
Coordinamento redazionale Raffaele del Monaco
Redazione Antonio Pasqua, Thomas Zaffino
Collaboratori S. Ascheri, M. Autiero, M. Bigatti, L. Buono, M. Camangi, A. Cangiano, M. Canducci, M. Casario, M. Del Gobbo, E. Florio, F. Grimaldi, A. Lippo, F. Lippo, R. Lombardo, A. Margarese, A. Marroccelli, M. Messina, G. Naccarato, C. Pelliccia, P. Perrotta, F. Sara, L. Spuntoni, E. Tavolaro, F. Vaccaro.
Segreteria di Redazione Veronica Longo

REALIZZAZIONE GRAFICA CROMATIKA S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Realizzazione Multimediale SET S.r.l.
Coordinamento Tecnico Piero Mannelli
Ideazione Grafica Gianluca Carbone
Realizzazione CD-Rom Paolo Iacona

PUBBLICITÀ Master Advertising s.r.l.
Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321612 - Fax 02 8321754
e-mail advertising@edmaster.it
Rete Vendita Serenella Scarpa, Cornelio Morari, Roberto Piano
Segreteria Ufficio Vendite Daisy Zonato

EDITORE Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321482 - Fax 02 8321699
Sede di Rende: C.da Lecco, zona industriale - 87030 Rende (CS)
Amministratore Unico: Massimo Sesti

ABBONAMENTO E ARRETRATI
Italia: Costo abbonamento annuale base (11 numeri) € 59,00 sconto 30% sul prezzo di copertina pari a € 84,70.
Costo abbonamento Plus (11 numeri + 6 libri) € 89,90, sconto30% sul prezzo di copertina pari a € 128,50.
Estero: Costo abbonamento annuale (11 numeri) € 169,40, costo abbonamento Plus (11 numeri + 6 libri) € 257,00.
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5,32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 028321482.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 028321699, oppure via posta a EDIZIONI MASTER via Cesare Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a: Edizioni Master Servizio Clienti - Via Cesari Correnti, 1 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
tel.02 8321482
@ e-mail: servizioabbonati@edmaster.it

Stampa: Elcograf Industria Grafica - Via Nazionale, 14 Beverate di Brivio (LC)
Stampa CD-Rom: Disctronics Italia S.p.a. Via G. Rossini, 4 Tribiano (MI)
Distributore esclusivo per l'Italia: Parrini & C S.p.A. Via Vitorchiano, 81 - Roma

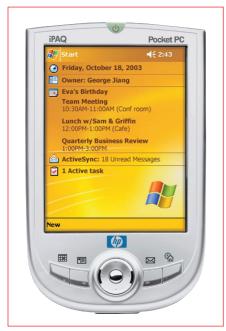
Finito di stampare nel mese di Luglio 2003

Edizioni Master edita:
Idea Web, Go!OnLine Internet Magazine, Win Magazine, PC Fun extreme, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all'Home Cinema, <tag/>, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, La mia videoteca, Le Collection.

News

Windows Mobile 2003 per Pocket PC

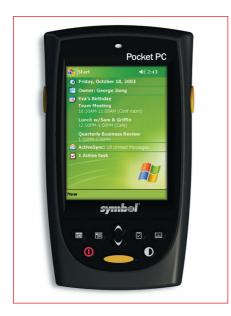
Wi-Fi e multimedia al centro della nuova piattaforma

La nuova versione di Windows Mobile è tutta tesa a semplificare e rendere più piacevole l'utilizzo del Pocket PC, attraverso un miglior supporto per l'accesso alle reti wireless, ai contenuti multimediali e ad una migliore integrazione con applicazioni e strumenti di sviluppo .NET. Il forte impegno di Microsoft è testimoniato dagli investimenti compiuti nel lancio del nuovo brand "Windows Mobile" che accompagnerà una nuova generazione di prodotti e strategie del gruppo.
Secondo la consolidata filosofia

Microsoft, la nuova piattaforma offrirà nuove opportunità in primo luogo ai produttori Hardware e Software, e porterà vantaggi anche ai gestori e produttori di reti wireless che potranno fornire nuove soluzioni per migliorare il modo di lavorare degli utenti.
Contestualmente al lancio della

nuova piattaforma, sono stati presentati due nuovi partner che entro la fine dell'anno lanceranno nuovi dipositivi: Gateway Inc. e JVC, che si vanno ad aggiungere alla lunga schiera di blasonati produttori. HP, Acer, Dell Toshiba, Fujitsu Siemens, Panasonic, sono solo alcuni dei nomi che stanno facendo grande il mercato dei Pocket PC.
Vediamo i principali miglioramenti apportati al nuovo Windows Mobile 2003:

- Connessione semplificata alle reti wireless, attraverso il rilevamento automatico di reti Wi-Fi ed il supporto nativo a Bluetooth.
- Una più efficace gestione della posta elettronica grazie alla integrazione con Microsoft Exchange Server 2003 (presto disponibile), che semplificherà i meccanismi di sincronizzazione della posta.
- Una più avanzata sezione multimediale che renderà più immediato l'utilizzo di musica e foto sui palmari e che, attraverso Windows Media Player 9, farà della riproduzione di video ad alta qualità uno dei punti di forza dei Pocket PC.
- Un più rapido turn over di nuove applicazioni, grazie al supporto integrato per il.NET Compact Framework, uno strumento che semplifica e velocizza lo sviluppo di nuovo software.

Windows Mobile 2003 si presenta, dunque, come una ideale piattaforma per lo sviluppo di nuove applicazioni che potranno trarre grande vantaggio dal nuovo sistema operativo Windows CE .NET 4.2 e dal .NET Compact Framework disponibili direttamente in ROM. Attraverso gli strumenti di sviluppo di Visual Studio .NET 2003, milioni di sviluppatori possono utilizzare la loro esperienza e le loro conoscenze per svi-

luppare innovative applicazioni mobili.
Un nuovo SDK per Windows Mobile 2003 è disponibile gratuitamente proprio per permettere agli sviluppatori di fare questo salto verso il mobile.

www.microsoft.com/mobile /developer/

Java e Linux alla conquista del pianeta rosso

Linux e Java alla conquista di Marte

Entro dieci anni pronto un veicolo per l'esplorazione del pianeta rosso. James Gosling ha presentato alla JavaOne di San Francisco un prototipo destinato alla esplorazione della superficie di Marte. Uno dei fattori chiave in queste missioni è la risposta in real time delle apparecchiature, che devono essere in grado di reagire senza esitazioni a qualsiasi sollecitazione. Il computer di bordo aveva infatti installato il sistema operativo TimeSys Linux RTOS e la JTime Real Time Java Visual Machine. Un grosso passo avanti per Java che va a sostituirsi alle applicazioni C/C++ che hanno sempre dominato nel settore delle mission critical.
Mars Exploration Rover, questo il nome del veicolo, è nato dalla collaborazione di Sun Microsystems con il Jet Propulsion Laboratory e sarà utilizzato nella missione Mars Expedition programmata per il 2009.

www.java.sun.com

Java incontra Python

Jython fa breccia nel muro che divide i due linguaggi

Nella comunità open source Python acquisisce sempre nuovi consensi, grazie alla sua semplicità e al fatto di essere rigorosamente object oriented e interpretato. Facile da imparare ed utilizzato ampiamente per lo scripting per Web Server ha dalla sua un forte supporto per XML e per la gestione delle immagini, ed è facile raggiungere risultati di rilievo con poche righe di codice. Quello che manca a Python è una reale diffusione, soprattutto in ambito business. Ora è forse possibile un salto di qualità: Jython è un interprete Python 100% pure Java, che consente di utilizzare le numerose librerie Python in progetti Java e, soprattutto, rende possibile la realizzazione di progetti misti, in cui è possibile affidare al linguaggio più adatto i compiti che di volta in volta vanno realizzati.

www.jython.org

Un server alla portata di tutti

Da Dell un server di facile gestione ad un prezzo molto interessante

La nota azienda, leader per la vendita diretta di computer e complete soluzioni informatiche, ha recentemente presentato un nuovo

prodotto rivolto prevalentemente alle piccole imprese. Il Dell PowerEdge 400SC è infatti un piccolo ma efficiente sistema, semplice da installare e gestire, ideale per svolgere la funzione di intranet server. E' possibile scegliere tra diverse configurazioni hardware, con CPU che vanno dal Celeron a 2 Ghz, fino al Pentium 4 a 3,2 GHz. Ma la caratteristica davvero interessante rimane il prezzo, infatti il modello PowerEdge 400SC IDE (CPU Intel Celeron 2 GHz, 128 MB di RAM, HD IDE da 40 GB, scheda di rete Gigabit) viene proposto al prezzo di soli ¤ 399 (iva e trasporto esclusi).

www.dell.it

A Francoforte vince Windows

Dopo il duro colpo di Monaco di Baviera, Microsoft si prende una rivincita sull'open source in tre città europee, fra cui Francoforte

Microsoft annuncia di aver vinto tre contratti governativi in altrettante grandi città europee, con lo scopo di spegnere l'entusiasmo con cui la comunità open source aveva

accolto, lo scorso maggio, la decisione del comune di Monaco di Baviera di migrare i propri sistemi informatici da Windows a Linux. Gli accordi permetteranno a Microsoft di installare su centinaia di computer dei comuni di Francoforte (Germania), Riga (Lettonia) e Turku (Finlandia) versioni desktop e server di Windows. Nonostante il dominio di Microsoft nel mercato aziendale e governativo in Europa, nell'ultimo periodo le soluzioni provenienti dal mondo del software open source si sono moltiplicate e sono riuscite, in vari casi, a spuntare alcune importanti commesse. Per questo, la vittoria in queste tre città appare significativa.

www.microsoft.it

Desktop Firewall 8.0

Una soluzione per la difesa dei desktop che include tecnologie di rilevamento delle intrusioni e funzionalità di firewall per pacchetti e applicazioni

McAfee Desktop Firewall 8.0 consente agli utenti enterprise di proteggere client remoti e fissi da minacce come codici maligni, hacker e applicazioni non autorizzate.
Il software include una nuova funzione di monitoraggio delle applicazioni per prevenire l'installazione o l'esecuzione di applicazioni non autorizzate. È disponibile inoltre una nuova modalità di quarantena che consente ad ePolicy Orchestrator di ispezionare il client prima che questo si colleghi alla rete.

Se l'antivirus del client è obsoleto o il firewall usa policy datate, l'accesso alla rete viene limitato.
La soluzione è già disponibile per le piattaforme Microsoft Windows 98 Se, Me, Nt 4.0, 2000 e Xp.

www.mcafeesecurity.com.

Districarsi fra le patch

Disponibile una guida da Microsoft per muoversi con disinvoltura fra le centinaia di aggiornamenti e patch disponibili

Microsoft ha pubblicato la prima versione della Guide to Security Patch Management, una guida che, come si legge sul sito, fornisce informazioni concise, suggerimenti, strumenti e template per aiutare le aziende a gestire, pianificare e applicare le patch di sicurezza sotto Windows 2000, XP e Server 2003.
La guida è disponibile solo in lingua inglese.

www.microsoft.com/downloads/

Apple presenta Safari 1.0

Apple ha annunciato l'introduzione sul mercato di Safari 1.0, a completamento del programma beta di ampio successo che ha fatto registrare quasi cinque milioni di download dal 7 gennaio 2003, data del lancio

Safari è diventato il browser preferito da milioni di utenti Mac e sarà il browser di default su tutti i nuovi computer Macintosh, a partire da Power Mac G5 annunciato oggi. Apple ha anche rilasciato un kit per lo sviluppo di software che consente agli sviluppatori di integrare il motore di rendering HTML di Safari direttamente nelle proprie applicazioni. Safari, il browser web più veloce mai creato per piattaforme Mac, porta una sferzata di energia alla categoria, grazie a innovazioni quali capacità di ricerca su Google integrate direttamente nella barra degli strumenti, SnapBack, un nuovo modo per tornare indietro all'istante ai risultati di una ricerca o al livello superiore di qualsiasi sito web dopo aver navigato nei livelli inferiori, un innovativo metodo per denominare, organizzare e presentare i preferiti, "tabbed browsing" e il blocco automatico delle finestre pubblicitarie a comparsa ("pop-up").
"Safari ha rinvigorito il settore dei browser introducendo funzionalità innovative che regalano la migliore esperienza web su qualsiasi piattaforma", ha dichiarato Steve Jobs, CEO di Apple. "Ogni utente Mac dovrebbe eseguire subito il download di Safari per unirsi ai milioni di utenti che lo stanno già usando".

www.apple.com

Lettore MP3/WMA e flash drive BenQ Joybee 120

Piccole dimensioni, massimo divertimento

BenQ presenta Joybee 120, il secondo dispositivo della serie MP3 BenQ Joybee. Con dimensioni compatte: 57 x 38 x 11,5 mm, Joybee 120 aggiunge un "tocco" di divertimento alla vita digitale grazie alle innumerevoli funzioni: MP3, WMA, stereo FM, registrazione vocale, FM e inoltre capacità flash drive USB. Piccolo nelle dimensioni, grande nel divertimento.
La dimensione ridotta e la compattezza del Joybee 120 insieme alla lunga durata delle batterie (~16 ore) sono caratteristiche progettate per chi è sempre in movimento.
La batteria agli ioni di litio e la pratica ricarica tramite una connessione USB a qualsiasi PC fanno del Joybee 120 il compagno perfetto da portare sempre con sé.
Il Joybee 120 ha in dotazione il software Q-Music di BenQ che permette di gestire facilmente la

musica che si preferisce e di modificarla secondo il proprio gusto, diventando un vero DJ. Il Joybee 120 consente inoltre di memorizzare fino a 60 brani MP3 da 4 MB circa, mentre il registratore incorporato può imprimere oltre 1000 minuti di audio, risultando una soluzione perfetta per riunioni, lezioni e annotazioni vocali. Dispone inoltre di firmware aggiornabile per consentire agli utenti di essere sempre al passo con le ultime novità del mercato.

Esprimi il tuo stile scegliendo tra uno dei tre straordinari colori disponibili: ghiaccio, verde e rosso bruciato.
Ora la vita digitale sarà veramente a colori. Il prezzo del Joybee 120 da 64MB è di euro 149 mentre quello da 128Mb è di 169.

www.benq.it

IIYAMA presenta la nuova gamma di LCD entry level

Due modelli con un nuovo design e assai completi dal punto di vista funzionale: pannelli dell'ultima generazione, veloci e luminosi, casse integrate, adatti per l'ufficio, così come per il gioco e le immagini in movimento

Con il lancio dell' E380S, lcd 15", si completa la nuova linea entry level di IIYAMA: una linea con un nuovo design molto accattivante, presentata in tre varianti colore (bianco, nero e silver/antracite). Anche le prestazioni sono estremamente interessanti: la matrice TN+Film consente di ottenere un'ottima luminosità sia per il 15" che per il 17" e offre un contrasto elevato e un angolo di visione che garantisce grande confort di utilizzo.

Entrambi i monitor prevedono la possibilità di impostare le modalità di risparmio energetico Eco 1 ed Eco 2: livelli ridotti di luminosità consentono di abbassare il consumo elettrico durante l'uso, di allungare la vita utile delle lampade ed ottenere un comfort di lavoro con tonalità di colore e flussi di luce meno aggressivi. L' E380S offre la funzione OPQ *(Optimal Picture Quality)* che consente, con la semplice pressione dell'apposito tasto in facciata, di variare luminosità e contrasto su una scala ottimizzata per la fruizione di immagini statiche, grafiche e per immagini in movimento. L'E380S ha le casse integrate ed è compatibile PC, Mac e workstation. Il prezzo al pubblico è di Euro 359,00 I.V.A. inclusa.

http://www.iiyama.it

SAGEM integra la tecnologia Java nei propri Telefoni Cellulari

Sagem, annuncia dei aver concluso il processo di integrazione della tecnologia Java nei propri telefoni cellulari

L'annuncio, rilasciato al termine di un periodo di ricerca e sviluppo congiunto tra SAGEM e Sun Microsystems, ha portato ad una completa integrazione della tecnologia JTWI *(Java Technology for the Wireless Industry)* negli apparati mobile della casa francese.
I telefoni cellulari SAGEM che integreranno la tecnologia JTWI, disporranno di tutte le funzioni *Java 2 Platform Micro Edition* (J2ME) *Connected Limited Device Configuration* (CLDC), *Mobile Informati on Device Profile* (MIDP 2.0), WMA e MMAPI.
Questa nuova piattaforma consentirà ai clienti SAGEM di avere accesso, grazie ad un ambiente aperto e standardizzato, sia ad applicazioni preinstallate o scaricabili, come i giochi, che a numerose altre applicazioni professionali o personali.
La tecnologia JTWI ri-definisce tutte le tappe principali legate alla prossima generazione di terminali Java.
L'iniziativa *"Java Technology for the Wireless Industry"* (JTWI), lanciata nel settembre 2002, è ormai entrata in una fase di sviluppo avanzata e accompagnerà i lanci di terminali compatibili JTWI. L'iniziativa JTWI conferisce una definizione chiara degli standard industriali e garantisce uno sviluppo costante della piattaforma destinata agli sviluppatori, consentendo agli operatori di lanciare più servizi legati alla trasmissione di dati e offrendo agli utilizzatori un ambiente più completo.
Integrando la versione 2.0 di MIDP nei propri prodotti, SAGEM utilizza l'ultima evoluzione degli standard Java per i telefoni cellulari, aumentando così la potenza e la portata delle applicazioni create per i telefoni cellulari: questa nuova versione dello standard Java migliora l'interfaccia utente, fornisce il supporto di funzioni sonore evolute, un'interfaccia di gioco che migliora le funzioni grafiche, la connettività all'Internet Mobile, e molte altre possibilità associate e funzioni che consentano di proteggere gli scambi e l'esecuzione delle applicazioni scaricabili.

http://www.sagem.com

JBuilder 9 Personal

Da Borland, la soluzione leader per lo sviluppo Java

Di ambienti di sviluppo per applicazioni Java se ne trovano veramente di tutti i tipi, ed ognuno di noi ha il suo preferito, che può variare anche a seconda della complessità del progetto nel quale si è impegnati. Quello trattato in questo articolo, che trovate in versione di prova allegato al CD-Rom della rivista, è uno tra i più imponenti e completi mai proposti, e và a rinnovare il concetto di programmazione in ambiente Java secondo la Borland. Sicuramente si tratta di un prodotto complesso, impegnativo per l'utente e per il sistema, che promette però enormi vantaggi sui tempi e le modalità di sviluppo, al prezzo di una curva di apprendimento delle funzionalità dello strumento piuttosto ripida. Ma scendiamo più nel dettaglio, analizzando quali sono le caratteristiche di punta del prodotto, e quali le novità che lo differenziano dal suo predecessore e dalle diverse versioni della stessa release.

Fig. 1: **Installazione.**

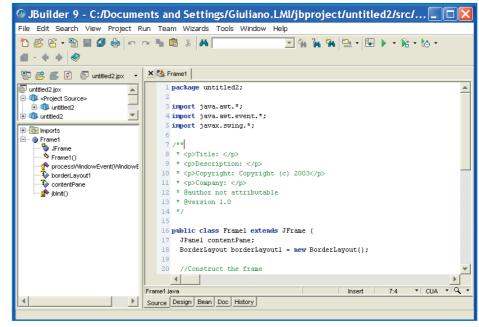
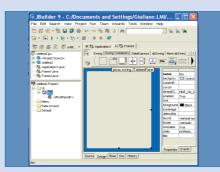
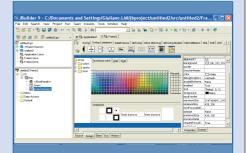



Fig. 2: **Schermata principale**

Programmazione visuale

1 **Per programmare un'interfaccia utente, il primo passo consiste nel creare un file che consenta questo tipo di operazione.**
Nella finestra che si apre quando selezionate *File/New*, indicate *Frame* come tipo di file da creare, oppure un altro dei file che possono contenere interfacce grafiche.

2 **Una volta in possesso del file in cui costruire l'interfaccia è necessario selezionare il pannello *Design* per poter iniziare a modellare l'interfaccia utente. Sulla parte superiore del programma sono presenti diversi pannelli carichi di oggetti da inserire nella propria applicazione è sufficiente selezionarli e tracciare un'area sulla maschera dell'applicazione.**

3 **Una volta creata l'interfaccia utente, se ne possono selezionare e modificare i componenti. Per selezionare un componente presente, si deve agire sull'albero collocato alla sinistra del programma, mentre, una volta selezionato, se ne possono modificare le proprietà impostandole direttamente nel pannello collocato alla destra.**

L'AMBIENTE DI SVILUPPO

L'ambiente di sviluppo integrato (IDE) JBuilder 9 si compone, come nel più classico dei sistemi software, di una finestra che contiene diverse aree dedicate alle possibili attività operabili sul progetto. Scrivere codice, disporre oggetti in maniera visuale, navigare tra i file di progetto e in un file in particolare, lanciare l'applicazione, eseguire il debbugging, compilare e creare l'applicativo è possibile senza lasciare mai l'interfaccia principale. Ma se cercavate un software leggero e immediato per modificare uno o due file java, vi accorgerete subito che questo non è lo strumento più adatto. Infatti tutto o quasi in JBuilder ruota attorno ai progetti che è necessario creare per accedere alle funzionalità avanzate, che sono la vera forza del prodotto. Per quel che riguarda l'area di text-editing, troverete tutte le funzionalità a cui i migliori programmi di sviluppo software vi hanno abituato, ed anche molto di più: possibilità di configurare completamente sia l'evidenziazione della sintassi, così come ogni parametro dell'editor, scorciatoie da tastiera, formattazione automatica del codice, completamento del testo, visualizzazione dei javaDocs e degli errori in linea, possibilità di utilizzo di template personalizzati nonché funzioni avanzate di ricerca e stampa. Per quanto riguarda la programmazione visuale di interfacce utente, sono presenti numerosi componenti, dai più utilizzati e conosciuti fino a quelli di presentazione di dati e connessioni ai database, con la possibilità di configurare completamente e visivamente ogni parametro dell'oggetto in questione e di integrare facilmente fra loro le varie parti per, costruire strutture complesse di interazione con l'utente. Tra l'altro sono presenti funzioni per la creazione, in modo intuitivo, di menu per le maschere dell'applicazione. Non meno utile risulta a livello di documentazione, la creazione automatica dei java-doc relativi alle classi ed alle funzioni create, che automatizza un processo troppo spesso trascurato della programmazione. Ma se lavorate in team, i soli java-doc potrebbero non bastarvi per tenere aggiornata e sotto controllo la situazione, e quindi ecco il pieno supporto a funzioni che visualizzano le informazioni sullo stato di modifica e di revisione di un file , nonché a prodotti di orientati allo sviluppo in gruppo come Microsoft Visual SourceSafe, Rational ClearCase ed altri.

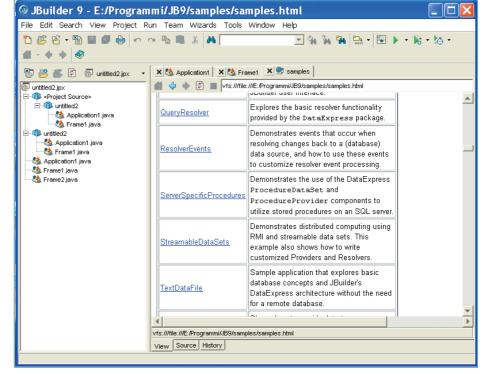
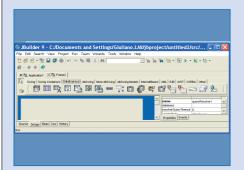



Fig. 3: Documentazione.

Supporto ai dati

1 JBuilder fornisce, oltre che le normali forme di connessione ai dati, dei componenti proprietari molto potenti che permettono di connettersi ed utilizzare basi di dati presenti sul proprio sistema. Per utilizzarli è possibile anche immetterli nell'interfaccia come un qualsiasi controllo, ed impostarne i le proprietà nel pannello delle proprietà

2 JBuilder disporre di un database integrato che tramite driver JDBC è possibile utilizzare per costruire le proprie applicazioni. Tramite *JDataStore* (questo è il suo nome), si possono creare nuovi database, impostarne le tabelle e riempirle di dati, nonché effettuare query e tutte le procedure di comune utilizzo. Per avviare *JDataStore* selezionare *Tool/JDataStore* Explorer.

3 Per configurare i parametri di *JDataStore* è possibile richiamare il server del servizio.
Per farlo è necessario selezionare *Tool/JDataStore Server*.
Da qui è possibile vedere lo stato dell' applicazione, impostare la porta su cui opera, visualizzare le connessioni e molto altro.

FUNZIONALITÀ AVANZATE

Le possibilità di JBuilder 9 sono veramente entusiasmanti, ed una volta appreso come possono semplificarvi la vita non credo che potrete privarvene facilmente. La versione Personal è quella dedicata allo sviluppo di applicazioni single-tier, per questo è quella in cui sono presenti in forma ridotta, ma nelle altre due versioni possibili *(Developer & Enterprise)* svolgono veramente la parte del leone. Pensate ad esempio alla possibilità di scegliere quale versione dell'JDK utilizzare per far girare l'applicazione che state sviluppando, alla capacità di suddividere il progetto in più parti per compilare o eseguire solo alcune parti, alla opportunità (nella versione Enterprise) di visualizzare l'applicazione sotto forma di diagrammi UML navigabili che semplifichino la visione d'insieme del progetto. Ma anche funzioni di refactoring avanzate, che vi permettono di aggiornare automaticamente dipendenze in caso di cambio di nomi, spostamenti, modifiche varie, rendendo di fatto un cambiamento al software un'operazione non preoccupante. Un'altra potente feature riguarda la funzione di debug, che permette di ispezionare in fase di runtime non soltanto i file locali dell'applicazione, ma anche, nella versione Enterprise del prodotto, servlet, pagine jsp, applet ed Enterprise Java Bean. Per chi sviluppa applicazioni Enterprise poi, esiste anche la possibilità di testare le prestazioni dei componenti distribuiti, per individuare eventuali colli di bottiglia e ottimizzare il funzionamento generale dell'applicazione (versione Enterprise). Se siete interessati dalle possibilità di integrazione del prodotto con le fonti di dati (e mi piacerebbe conoscere chi non lo sia oggigiorno), sarete contenti di sapere che la Borland vi semplifica il lavoro permettendovi di disporre di componenti aggiuntivi specifici per questo scopo, che rendono immediata la presentazione dei dati e la modifica di questi ultimi, sia sotto forma di componenti di interfaccia utente, che come funzioni di data-browsing integrate nell'interfaccia principale. JBuilder 9 prevede poi la possibilità di distribuire il proprio lavoro, inserendo in un unico pacchetto i file che compongono la propria applicazione, consentendo di distribuire il software in diversi formati, che spaziano dal classico eseguibile, all'applet, passando per gli Executable Jar e diversi altri. Ovviamente, essendo Java pensato per funzionare senza modifiche su tutte le piattaforme, è possibile costruire, a partire dallo stesso codice e grazie al comodo wizard, applicativi per tutti i maggiori sistemi operativi: Windows, Linux, Solaris e Mac OS X.
Write once, run everywhere!!

PER CONCLUDERE

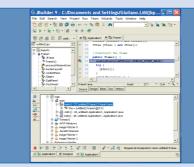
In conclusione, come già anticipato, questo prodotto è veramente completo, ed anche se la versione personal risulta essere quella indirizzata allo sviluppo di applicazioni non distribuite, e quindi la meno complessa, è comunque adatta ad un utilizzo di tipo professionale. Se da poco vi siete cimentati con lo sviluppo di applicazioni Java, forse sarebbe preferibile un prodotto più semplice che vi permetta di spendere più tempo nella scrittura di codice piuttosto che con le funzioni della sua interfaccia.
Se invece siete padroni del linguaggio, e siete alla ricerca dell' IDE definitivo, che permetta di compiere operazioni avanzate e che garantisca automazioni che facilitino e velocizzino lo sviluppo, beh allora credo che siate arrivati nel posto giusto. Questo strumento si posiziona infatti ai massimi livelli qualitativi, e risulta essere la scelta migliore per coloro che hanno esigenze elevate o progetti di media o grande complessità da sviluppare.

Giuliano Uboldi

Requisiti di sistema

- **256 MB di memoria RAM (512 Raccomandati)**
- **500 MB di HardDisk liberi**
- **CD-Rom Drive**
- **Monitor ad alta risoluzione (1024 x 768 - 256 colori minimo)**
- **Dispositivo di puntamento (mouse o trackball)**
- **Intel Pentium II/233 MHz o superiore (o compatibile)**
- **Microsoft Windows 2000 (SP2) , XP, NT 4.0 (SP6a)**

Correzione degli errori

1 **JBuilder dispone di diversi modi per assicurarsi che la propria applicazione contenga meno errori possibili. Il primo è la correzione automatica, che segnala in tempo reale la presenza di errori di sintassi o anche nomi di classi o variabili sconosciute. Non sarà necessario compilare il programma per conoscere eventuali errori di digitazione.**

2 **In caso di errori più seri occorre utilizzare il debugger. Per avviare il processo occorre selezionare, dopo aver impostato dei *BreakPoint* facendo click con il tasto destro sulla riga di codice che intendiamo controllare, ed avere selezionato la voce adatta, *Run/Debug Project*. Si avranno a disposizione tutti gli strumenti per controllare passo passo l'esecuzione.**

3 **Se le cose si dovessero poi fare veramente difficili, è sempre possibile ricorrere all'help del prodotto, che integra oltre che alla documentazione ufficiale *JavaDoc* e alla documentazione sul programma, anche esempi e tutorial che è possibile scaricare gratuitamente dal sito della Borland, a patto di disporre di una connessione ADSL.**

Macromedia Authorware 7

La prima scelta per la crezione di contenuti e-Learning.

Sempre all'avanguardia nel settore della programmazione di applicazioni multimediali, con questa nuova versione di Authorware, Macromedia ha ampliato notevolmente le potenzialità dell'ambiente attraverso numerosi miglioramenti. Tra i tanti, segnaliamo: la possibilità di importare presentazioni Power Point, il supporto per la scrittura e l'esecuzione di codice JavaScript, la gestione di XML sia in input che in output, il pieno supporto per il formato DVD ed una nuova e più intuitiva interfaccia utente. Proprio grazie alla nuova interfaccia, Authorware può proporsi con eguale forza sia agli sviluppatori esperti sia a programmatori alle prime armi o utenti occasionali: a tutti è garantita la possibilità di sviluppare software di e-Larning al massimo livello. Authorware 7 consente di integrare qualsiasi contributo (grafica, audio, animazioni, video e filmati Flash) all'interno di un ambiente coerente con l'impostazione di tutta la famiglia Macromedia MX, con un ampio uso del drag&drop per semplificare la creazione delle applicazioni. Fondamentale risulta la possibilità di generare applicazioni sulla base di presentazioni Power Point già esistenti. Il diffuso supporto di XML consente poi di creare applicazioni estensibili i cui comportamenti e contenuti possono essere facilmente modificati anche in fase post-deployment.

Fig. 1: Wizard per la creazione di applicazioni.

LO SVILUPPO DELL'APPLICAZIONE

Ogni volta che creiamo una nuova applicazione, ci viene chiesto il tipo di contenuto che vogliamo produrre. Da questo punto in poi saremo guidati da un wizard alla creazione degli elementi fondamentali dell'applicazione. L'intero processo di sviluppo avviene attraverso una interessante commistione di programmazione visuale e codice, che consente di tenere sempre sotto controllo l'intera struttura dell'applicazione. Questa caratteristica, unita alla assoluta uniformità allo standard Macormedia per le interfacce utente, riduce moltissimo il tempo di apprendimento. Tutto l'ambiente risulta ampiamente configurabile con pannelli e barre di comando aggregabili e riposizionabili a piacere.

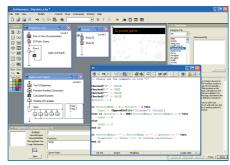



Fig. 2: L'ambiente offre una interessante combinazione di componenti visuali.

LE NUOVE CARATTERISTCHE

Il nuovo supporto per i DVD consente di aumentare notevolmente la durata e la qualità delle presentazione e dei contenuti che si vogliono proporre. Inoltre, il supporto a JavaScipt (ottenuto grazie allo stesso motore presente in Dreamweaver MX) consente una fortissima dose di interattività, che si traduce in applicazioni più divertenti ed efficaci. Infine, il supporto ai dati XML consente una più semplice connessione a contenuti Web, migliorando l'esperienza degli utenti e semplificando la gestione e l'aggiornamento dei prodotti distribuiti.

NON SOLO TRAINING

Benché il campo d'azione preferito sia dichiaratamente l'e-learning e le applicazioni di training, Authorware si presta ottimamente alla creazione di qualsiasi applicazione multimediale. Presentazioni, giochi e chioschi virtuali sono un piccolo esempio di quello che l'ambiente può realizzare. La completa apertura verso Power Point e verso tutti gli altri strumenti Macromedia, fa il paio con il forte supporto per XML, rendendo possibile armonizzare Authorware in qualsiasi contesto lavorativo. Macromedia Authorware 7 è disponibile sia come prodotto a se stante, sia come parte della Macromedia eLearning Suite, che include anche Flash MX e Dreamweaver MX.

Fig. 3: Pieno supporto per l'XML.

Installazioni

In alcuni casi si sono manifestati dei problemi nella fase di installazione dell'applicativo. E' possibile che, dopo la procedura di installazione, al primo tentativo di avvio, si riceva un messaggio di errore. In questo caso è sufficiente lanciare il file VSetupT.exe, locato nella directory di installazione di Authorware (tipicamente C:\Programmi\Macromedia\Authorware 7.0). Fatto ciò non ci dovrebbero presentarsi altri problemi e l'applicazione sarà pronta ad essere avviata correttamente.

Authorware 7

- **Produttore: Macromedia**
- **Licenza: Trial**
- **Prezzo: € 2929 ***
- **Aggiornamento: € 389 ***
- **Prezzo della eLearning Suite: € 3419 ***
- **Sul web: www.macromedia.it**
- **Sul CD: \soft\tools\Authorware**

i prezzi sono al netto dell'IVA

Flash IntroBuilder-SX Lite 3.0

Per creare introduzioni Flash in un lampo!

Un potente e semplice tool che consente di creare affascinanti introduzioni Flash ai nostri siti in pochi istanti e con uno sforzo pari a zero! Un semplicissimo Wizard ci guida nella scelta dello sfondo, dell'animazione e della musica che accoglierà i visitatori delle nostre pagine Web. Una presentazione in Flash del proprio sito è un po' la carta da visita del nostro lavoro: considerate superflue e noiose fa molti, le introduzioni possono comunque rappresentare un'attrattiva ed un modo per esprimere efficacemente lo "spirito" del sito che abbiamo sviluppato.

Fig. 1: Un esempio dei risultati ottenibili con Flash IntroBuilder-SX.

FACILE

Il lavoro di chi si accinge a creare un'introduzione con IntroBuilder si riduce alla scelta delle varie componenti multimediali: le musiche (in Mp3) da utilizzare come sottofondo, le immagini delle scene che comporranno il filmato, l'animazione di presentazione. Il rischio di omologazione che si corre adottando prodotti "precotti" come questo è mitigato dall'ampia scelta di scene e combinazioni possibili. Una volta effettuate tutte le scelte richieste, è sufficiente cliccare sul pulsante *"Save Current Project"* per salvare il file Flash di introduzione. Per utilizzarlo nel nostro sito, è necessario poi effettuare l'upload del file ottenuto nella root del Web Server. Non è esclusa la possibilità di copiare il file su CD ed utilizzarlo in locale per presentazioni multimediali. La scintillante bellezza delle animazioni Flash è ora disponibile a chiunque, senza che sia richiesto alcun grado di esperienza.

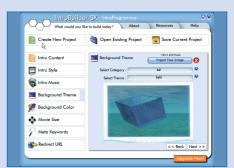
Fig. 2: Splash-screen dell'applicazione.

IntroBuilder-SX Lite 3.0

- **Produttore:** WebDesignHQ.com
- **Licenza:** trial
- **Prezzo:** $199
- **Sul Web:** http://www.webdesignhq.com/
- **Sul CD:** \soft\tools\IntroBuilder\

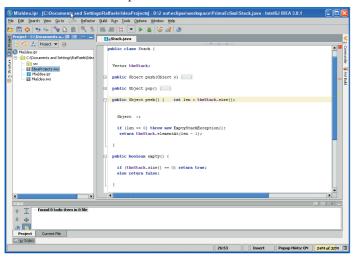
Pochi passi per creare un'introduzione

1 Tutta la creazione dell'introduzione è completamente assistita da wizard.
Si comincia con la scelta della directory e del nome del progetto.
L'impatto visivo contribuisce a rendere più piacevole lo sviluppo.

2 E' possibile scegliere fra decine di stili,ognuno caratterizzato da animazioni molto curate. L'effetto generale ha un aspetto decisamente "cool", e la vastità dei temi proposti consente di integrarsi con coerenza in qualsiasi sito.

3 I numerosi effetti sonori e gli sfondi che è possibile utilizzare, sono stati scelti con estrema cura. La resa finale è un introduzione di altissimo livello, soprattutto se si considera il tempo davvero ridotto che ci è richiesto.

IntelliJ Idea 3.0.4

Sviluppare in Java… con piacere!

Idea è un ambiente di sviluppo che consente di creare applicazioni Java all'interno di un IDE ben fatto e gradevole. In questa nuova versione, aggiunge nuove importanti caratteristiche: pieno supporto per JSP/EJB, integrazione con Ant e JUnit, supporto per XML e una cospicua collezione di API per la realizzazione di plug-in. Inutile sottolineare che sono presenti tutte le più importanti caratteristiche comuni in un moderno IDE: un ottimo debugger, avanzate funzioni di search&replace, colorazione sintattica. Una menzione particolare la merita il completamento automatico del codice che si fa apprezzare con l'uso per la sua semplicità e per la flessibilità con cui segue le abitudini dello sviluppatore. È possibile personalizzare qualsiasi aspetto dell'applicazione: dalla formattazione del codice, alla colorazione sintattica, all'organizzazione dell'import, fino ad arrivare ai colori utilizzati per evidenziare gli errori e all'orientamento delle finestre e delle toolbar. Un'assenza di peso risulta essere quella di un apposito strumento per la realizzazione dell'interfaccia utente. A differenza di molti altri IDE, Idea ha coraggiosamente rinunciato a questa caratteristica che, sebbene comoda in molte situazioni, risulta spesso un inutile appesantimento per i programmatori che non ne fanno uso.

Nota

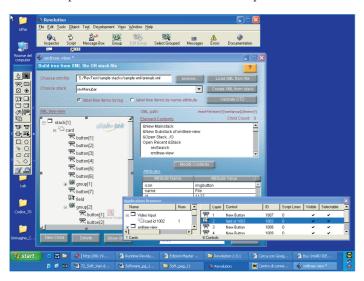
Per richiedere la chiave di attivazione è necessario compilare un form all'indirizzo: http://www.intellij.com/download/idea/try/windows. L'indirizzo è diverso da quello proposto all'atto dell'installazione di IDEA.

IntelliJ Idea 3.0.4

- **Produttore: JetBrains, Inc.**
- **Sul Web: www.intellij.com/jetbrains/**
- **Prezzo: $499**
- **Sul CD: \soft\tools\IDEA**

Revolution 2.0.1

Per creare applicazioni multipiattaforma

Per gli sviluppatori che vogliono creare applicazioni che possano girare su qualsiasi sistema operativo da Mac OS a Windows, passando per il sempre più popolare Linux: ecco pronto Transcript, un linguaggio di alto livello, integrato in Revolution, e che consente di sviluppare applicazioni efficienti in un tempo brevissimo rispetto a linguaggi più classici come Java, C++ o VB. Due caratteristiche su tutte: multipiattaforma, e brevità. Funzioni che prima richiedevano complesse subroutine, si risolveranno con una singola riga di codice. La versione che trovate nel cd limita ad un massimo di 10 righe il codice associabile ad ogni oggetto. A parte questa limitazione, troverete tutte le funzioni disponibili nella versione a pagamento e potrete anche distribuire le applicazioni sviluppate con questo pacchetto. In questa versione è stato integrato il supporto per SOAP ed una libreria specifica per il parsing XML. Tra le cose che più si apprezzano è che l'applicazione non necessita di installazione e non va ad incidere sul registro di sistema: è sufficiente compattare il file compresso e lanciare l'exe di riferimento. Al primo avvio ci verrà chiesto se vogliamo l'esecuzione in modalità gratuita o in modalità evaluation. La modalità gratuita non ha limiti di tempo e supporta unicamente la connessione via ODBC ai database. La versione evaluation, oltre ad ODBC, supporta nativamente Oracle, MySQL e PostgreSQL, ma scade dopo trenta giorni. Per una più completa informazione sulla licenze disponibili si rimanda al web: *http://www.runrev.com/Revolution1/licensing1.html.*

Revolution 2.0.1

- **Produttore: Runtime Revolution Ltd.**
- **Sul Web: www.runrev.com**
- **Prezzo: free**
- **Sul CD: \soft\tools\revolution.zip**

JBoss 4.0.0DR1

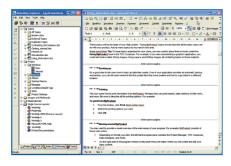
Il più potente application server open source

JBoss è un application server Open Source scritto interamente in Java. Grazie a JBoss è possibile far girare componenti EJB (Enterprise Java Beans) e qualsiasi applicazione sviluppata attraverso la tecnologia J2EE di Sun.
Grazie ai connettori JMX, JBoss consente una notevole flessibilità e si presta agli utilizzi più disparati. In particolare è possibile sviluppare un proprio transaction-manager ed un proprio persistence-manager.
Nella versione che trovate nel CD è incluso Tomcat 4.1.24 al posto di Jetty

Sul CD: \soft\tools\Jboss

RoboHelp Office X4

Creare Help di livello professionale

Un tool di altissimo livello professionale per la creazione di help sia per applicazioni desktop che Web Based. RoboHelp è in grado di creare sistemi di help utilizzabili attraverso qualsiasi browser e su qualunque piattaforma. Le facilitazioni per lo sviluppatore sono molte e vanno dalla creazione in automatico di indici e glossari, alla generazione di oggetti grafici.

I contenuti possono essere importati a partire da numerosi formati, inclusi HTML e Word.
Versione di valutazione valida quindici giorni.

Sul CD: \soft\tools\robohelp.exe

FsDBPrint 1.0

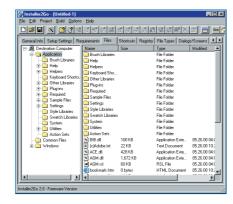
Per stampare la struttura delle tabelle di Access

Alzi la mano chi non ha mai fatto uno screenshot della struttura delle tabelle di un database in Access, al fine di documentare o solo tenere traccia delle relazioni impostate.
Questa piccola ma utilissima applicazione ci solleva proprio da questo umiliante(!) compito: i documenti stampati risultano al contempo facili da leggere e compatti. E' possibile stampare anche il dettaglio dei campi che compongono le varie tabelle.

Sul CD: \soft\tools\fsDBPrint.exe

Installer2Go Freeware 3.1.1

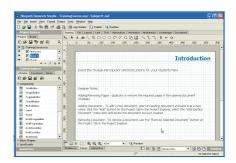
Un sistema facile e gratuito per realizzare pacchetti di installazione

Un tool per la creazione di file autoinstallanti che non impegna lo sviluppatore nel dover imparare l'ennesimo linguaggio di scripting: con una interfaccia che punta tutto sul drag&drop, realizzare pacchetti di installazione diventa un vero gioco da ragazzi.

Benché gratuito, Installer2Go va incontro a tutte linee guida per la certificazione WindowsXP.

Sul CD: \soft\tools\i2g1.exe

Norpath Elements Studio 2.0

Creare applicazioni multimediali e interattive senza scrivere codice

Norpath Elements Studio è un innovativo ambiente per la creazione di applicazioni multimediali che rinuncia completamente al codice a favore di una programmazione completamente visuale.
Un completo supporto per elementi multimediali, animazioni e database, ne fa un ottimo strumento per la realizzazione di applicazioni per l'e-lear

E' possibile distribuire le applicazioni su una pluralità di media: dai CD-ROM, ai chioschi multimediali alle intranet.
Versione di prova valida venti giorni.

Sul CD: \soft\tools\nestudio-win.exe

EmEditor 3.34

Un editor pronto a qualsiasi linguaggio

Un versatile editor testuale che, grazie al supporto per i plug-in, può crescere insieme alle nostre esigenze. Molti sviluppatori l'hanno già scelto grazie soprattutto all'ampio supporto fornito per tutti i più diffusi linguaggi di programmazione:

ASP, C++, C#, CSS, HTML, Java, JavaScript, JSP, Pascal (Delphi), Perl, PHP, Python, Ruby, SQL, Tex (LaTeX), VBScript, e Windows Script.

Ognuno di questi linguaggi gode di un proprio syntax highlighting che semplifica la lettura di codice scritto da terze parte e riduce la possibilità di commettere errori durante la digitazione di nuovo codice.
Versione di prova valida trenta giorni.

Sul CD: \soft\tools\eme334e.exe

SQLSchema 2.2.2

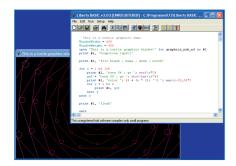
Compara la struttura di due database SQL Server

Un tool per amministratori di basi di dati che consente di confrontare lo schema di due database SQL Server. Oltre a confronto, SQLSchema consente di allineare le differenze esistenti fra i due database.
Tutti i cambiamenti effettuati sono salvati in un file di Log.
Versione dimostrativa.

Sul CD: \soft\tools\sqlschema.2.2.2.setup.exe

Liberty BASIC for Windows 3.03

Un linguaggio per cominciare bene

Un linguaggio di programmazione sviluppato appositamente per chi si avvicina la mondo della programmazione.
Grazie alla funzione FreeForm si possono sviluppare complesse interfacce Windows con la semplicità offerta da un ambiente visuale.
L'ide comprende un editor con funzioni di highlighting sintattico e un completo debugger.

Versione di prova limitata a trenta giorni.
Sul CD: \soft\tools\lb303win.exe

3D Developer Studio for Visual Basic 6.0

Crea il tuo videogioco in Visual Basic

Un ambiente per realizzare applicazioni in 3D utilizzando Visual Basic 6.0. Utilizzato da molti sviluppatori per l'alto livello qualitativo raggiungibile con un ridotto sforzo ed in brevissimo tempo. Indirizzato principalmente alla creazione di videogiochi, può essere proficuamente utilizzato per realizzare applicativi tridimensionali di qualsiasi genere. La licenza è gratuita, per l'installazione è necessaria una chiave di attivazione che si può richiedere collegandosi all'URL: *http://www.3dstate.com/Pages/PagesI/Registration/downloadlist.htm*
Sul CD: \soft\tools\3D_Developer_Studio_VB602.exe

checkstyle 3.1

Il nostro codice sempre al meglio

Checkstyle è principalmente un tool a linea di comando che esegue un controllo di conformità su un file Java (o un gruppo di file fino ad un intero progetto grazie anche alla completa integrazione con ANT). Le regole da verificare ed il grado di severity di una eventuale non conformità sono configurabili e memorizzabili in file XML per i quali è presente un'apposita grammatica.
Insieme al pacchetto sono disponibili già, come esempio, due file di configurazione pronti per l'uso: uno contenente un pacchetto di regole personalizzato di Checkstyle ed un altro contenente una completa rimappatura per Checkstyle delle Code Conventions.
Sul CD: \soft\tools\checkstyle-3.1.zip

Sirid 1.14

Per gestire i tuoi progetti software

Un sistema professionale per la gestione di progetti software: con Sirid è possibile gestire il ciclo di vita di un'applicazione dal punto di vista dello sviluppatore.
E' possibile assegnare compiti e priorità ai componenti del team ed i bug riportati, oltre ad essere memorizzati in un database, possono essere automaticamente inoltrati al responsabile del progetto.

Grafici statistici e timesheet aiutano ad ottimizzare il lavoro degli sviluppatori.
Sirid è accessibile attraverso un comune browser, non è dunque necessario alcuna installazione lato client e risulta garantita la delocalizzazione della forza lavoro.
Sul CD: \soft\tools\sirid.exe

602Pro LAN Suite 2003

Un mail server gratuito e sicuro

Un mail server gratuito, che integra funzioni anti-virus, anti-spam e firewall. La sicurezza delle comunicazioni è garantita dall'uso del protocollo SSL SMTP/POP3, mentre per l'antispam è possibile definire una lista nera ed è garantito l'aggiornamento tramite DNS.
Sul CD: \soft\tools\ls2003.exe

HTML Ease Pro 2.0

I tuoi documenti sul Web... con un clic!

Un avanzato ambiente per la preparazione di pagine Web. HTML Ease è in grado di convertire in pagine html qualsiasi documento prodotto dai più diffusi programmi di Word Processing e fogli elettronici.
E' incluso un completo editor ricco di funzionalità per semplificare la scrittura di codice HTML.
Versione di prova limitata a dieci giorni.
Sul CD: \soft\tools\hep20b.exe

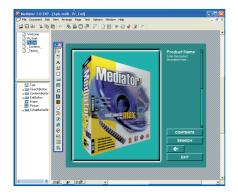
Apache Axis

Un SOAP engine in Java

Axis è un SOAP Engine e consiste in un'implementazione Java della specifica SOAP.
Si tratta della terza generazione di Apache SOAP ed è curata e rilasciata in modalità open source dalla Apache Software Foundation all'interno dell'ambizioso Apache XML Project.
In questa versione Axis è stato profondamente ristrutturato e mentre prima si trattava di un semplice strato software in grado di renderci trasparente l'utilizzo di SOAP nello sviluppo di applicazioni scritte con tecnologia Java, adesso è un vero e proprio framework che ci permette di costruire, con estrema semplicità, tutto quello che ci può servire per rendere fruibili i nostri servizi sotto forma di Web Services: server, client, proxy, stub, skeleton e gateway da e verso il protocollo SOAP.
Ci troviamo di fronte quindi ad un ambiente completo che garantisce l'interoperabilità con servizi che utilizzano tecnologie differenti grazie al supporto completo e nativo per WSDL 1.1.
Nella directory trovate anche i sorgenti.
Sul CD: \soft\tools\Axis

Mediator 7 exp

Creare le tue presentazioni per qualsiasi media

L'intuitiva interfaccia di Mediator (completamente drag&drop) unitamente alla flessibilità data dal supporto per la programmazione in VB e JavaScript, consente di creare ricchissime applicazioni multimediali in poco tempo. Particolarmente indicato per la realizzazione di presentazioni di livello professionale o per applicazioni di training che prevedono una forte interattività con gli utenti.

Versione dimostrativa, non sono previsti limiti di tempo per il suo utilizzo, ma le applicazioni create hanno un limiti di sette giorni.

Sul CD: \soft\tools\m7uk_exp_demo.exe

DzSoft Perl Editor 5.4

Per la scrittura e il debug di script in Perl

Un editor che presenta tutte le principali funzioni per scrivere ed effettuare il debug di script PERL. Oltre al consueto Syntax highlighting, ci ha ben impressionato la possibilità di testare il codice senza la necessità di lanciare un Web Server.

È possibile uploadare il codice via FTP direttamente dall'ambiente.

Sul CD: \soft\tools\dzperl54.zip

Advanced Hex Editor 3.2

Sviluppo e recupero dati a basso livello

Un editor che permette di analizzare e modificare a basso livello dati contenuti sia su disco sia in memoria.

Utilissimo per il reverse engineering e soprattutto per le situazioni di recupero dati. Ricco di funzionalità per la gestione di grosse porzioni di dati, prima fra tutte una funzione di ricerca ottimizzata.

Versione di prova valida trenta giorni. rop; CRC generation; diffing; and more.

Sul CD: \soft\tools\installAXE.exe

RenderX XEP 3.5

Per convertire documenti XML in PDF

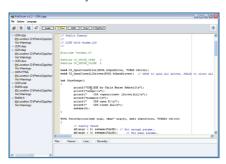
Un'applicazione commerciale in grado di effettuare la conversione da documenti XML in PDF o in formato PostScript. Accetta in input dati in formato XML e fogli di stile XSL, il rendering è effettuato proprio come combinazione dei due ingressi.

Versione dimostrativa, un watermark è applicato su ogni documento prodotto. XEP offra un valido supporto a XSL FO, grafici SVG e funzioni di link PDF-to-PDF.

Sul CD: \soft\tools\xep35_trial.zip

RatScan 1.2

Testa la robustezza del tuo codice

Attraverso una efficace interfaccia grafica, RatScan consente di esaminare il nostro codice alla ricerca di difetti e vulnerabilità E' possibile eseguire lo scan su codice scritto in PHP, C/C++, Perl, e Python.

Versione di prova limitata a trentuno giorni.

Sul CD: \soft\tools\RatScan_1.2.exe

TestTrack Pro 5.1

Tracciare bug e richieste degli utenti

TestTrack è un tool cross-platform per la catalogazione di bug e la pianificazione dello sviluppo software. Può essere interrogato via browser e si integra perfettamente con Visual Studio 6, Visual Studio .NET e Visual SourceSafe. Permette agli utenti Windows ma anche Macintosh di accedere simultaneamente allo stesso database cross-platform.

Inoltre, include ottime funzioni per il filtering permettendo di creare delle liste di problemi organizzate per priorità. Ottima la possibilità di generare dei report.

Licenza di valutazione della durata di trenta giorni, è possibile tracciare un massimo di 15 bug.

Sul CD: \soft\tools\ttprowininstall.exe

OraEdit 3.3

Un editor su misura per Oracle.

La versione aggiornata dell'ottimo editor gratuito progettato su misura per gli sviluppatori Oracle. Il tool permette di editare e compilare il codice PL/SQL direttamente dal database, consente di creare nuovi oggetti a partire da templare, presenta funzioni di import/export del codice ed altre utili caratteristiche.

OraEdit presenta il supporto per i workspace e consente di lavorare su più database allo stesso tempo.

La versione 3.3 include il pieno supporto per Oracle 9i, nuovi menu in stile XP, una interfaccia di più veloce utilizzo e numerosi altri ritocchi.

Sul CD: \soft\tools\oraedit.exe

Surround SCM 1.5

Tiene traccia dei cambiamenti durante lo sviluppo delle applicazioni

La gestione dei cambiamenti apportati durante lo sviluppo delle applicazioni è una componente essenziale per lo sviluppo di qualsiasi applicazione.

Grazie alla sua integrazione con Visual Studio, Surround SCM consente di velocizzare le operazioni di gestione, migliorando la qualità del codice che scriviamo.

Versione di valutazione valida trenta giorni.

Sul CD: \soft\tools\sscmwininstall.exe

L'arte di nascondere le informazioni.

Algoritmi Steganografici

La steganografia, insieme alla crittografia, è uno dei fondamentali strumenti per la trasmissione di messaggi segreti ed è quindi usata in modo massiccio nell'ambito della sicurezza informatica.

Abbiamo già trattato l'argomento steganografia nello scorso appuntamento, evidenziando le fondamenta teoriche ed approfondendo alcuni importanti risvolti di carattere tecnico. Tra quelle pagine avevo anche lanciato una sfida, al momento in cui vi scrivo, tale sfida non è stata raccolta da nessuno, o almeno nessuno è riuscito a risolvere il problema proposto. A tale proposito ricordo che avevo inserito nel testo dell'articolo del mese scorso un "semplice" messaggio steganografato. Chi ha perso il precedente numero di ioProgammo si starà chiedendo di cosa stiamo parlando. Brevemente tenterò di introdurre nuovamente il problema, dando questa volta nuovi spunti, ma soprattutto, presentando degli algoritmi risolutori attualmente utilizzati. La steganografia è un metodo per nascondere (a rigore seguendo l'etimologia della parola greca si dovrebbe parlare di occultare) un qualcosa all'interno di un altro qualcosa. Ma cosa sono questi due qualcosa. Nell'accezione più generale possono essere elementi molto astratti e generici. Abbiamo visto come in un apparentemente innocuo discorso pronunciato da chiunque, si dice ad esempio che alcuni terroristi ne facessero uso compreso Bin Laden, possa esserci innestata una frase segreta. Certo, nell'ambito che ci riguarda gli elementi in gioco sono informazioni in formato digitale e quindi: testi, immagini, suoni e tutto ciò che possa essere memorizzato come un generico file. Così, una foto di amici che a prima vista non insospettirebbe affatto, potrebbe celare importanti messaggi o altri immagini segrete.

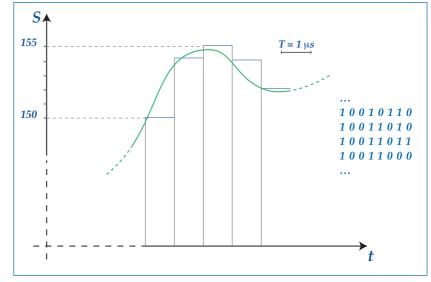



Fig. 1: Campionamento a fc di 1KHz e quantizzazione a 8 bit, di una parte di segnale.

O ancora un file sonoro che riproduce una canzone in voga può nascondere testi o altre informazioni segrete. Gli elementi in gioco, qualunque essi siano, per comodità saranno indicati in seguito come messaggi. La potenza del metodo è di trattare elementi digitali apparentemente innocui, non soggetti quindi a controlli. Nella crittografia, invece, si sa benissimo a priori che si ha un messaggio cifrato, la difficoltà nel caso specifico è decifrarlo. L'unione dei due metodi assicura elevatissimi gradi di sicurezza nella trattazione di messaggi segreti. Ma entriamo nei particolari esaminando alcuni metodi e algoritmi.

METODO LSB, DI COSA SI TRATTA

Il metodo studiato nella puntata precedente è LSB. Lo riesaminiamo brevemente poiché esprime al meglio la filosofia sottesa alle tecniche steganografiche. Chi ha letto l'articolo del numero scorso può saltare il presente breve paragrafo. Si parte dal presupposto che un'immagine, come un file sonoro, rimane pressoché

Soluzioni

Campionamento e quantizzazione

Il campionamento e la quantizzazione sono i due passi da attuare per discretizzare un segnale analogico. Permettono cioè di trasformare una qualsiasi curva, che ad esempio descrive un segnale sonoro originale, in una sequenza di bit. Tale trasformazione è indispensabile per la trasmissione e il trattamento di segnali, poiché come l'abc del programmatore ricorda l'elaboratore lavora solo sequenze di bit. La frequenza di campionamento (fc) indica per quali intervalli di tempo (T) bisogna misurare il segnale, si ricorda la relazione fc=1/T, mentre i range di quantizzazione, che saranno maggiori a fronte di un numero più elevato di bit, approssimano il valore della curva ad un numero che successivamente sarà trasformato in una sequenza di bit (ad esempio, 8 o 16 bit). In figura è mostrato il processo di campionamento e quantizzazione (discretizzazione) di un segnale analogico.

inalterata, o comunque subisce variazioni impercettibili, qualora si "manomettano" i bit meno significativi delle informazioni di base. Ad esempio, un file bmp (bit map) rappresentato da una mappa di pixel di colore diverso è costituito da sequenze di terne di byte. Secondo il metodo RGB (red, green, blue) ogni pixel assume il proprio colore come l'unione di tre primari, appunto rosso, verde e blu, la cui intensità di ognuno può variare in una scala tra 256 valori, ossia tutte le possibili configurazioni che può assumere un byte. Secondo tale modello ad un pixel sono associati tre byte, altri modelli prevedono un numero maggiore di byte. Cambiando l'ultimo bit di alcuni dei byte che formano la tinta, viene alterato il colore del pixel in modo impercettibile. Si pensi che di per se il pixel è piccolo, è l'unità minima grafica, un semplice punto, inoltre, si altera il contributo di un colore facendolo passare in un livello di intensità contiguo al precedente in una scala di 256. La sequenza dei bit così modificati da vita ad un'altra immagine, quella segreta, che in tal modo viene iniettata nella immagine principale denominata di copertura. Analoga conseguenza si raggiunge se il file iniziale di copertura è un sonoro o un filmato. Tale metodo fa comprendere come funziona la steganografia, ad ogni modo non è molto efficiente.
Ad esempio, si rischia di perdere il messaggio segreto se l'immagine di copertura viene modificata (ridimensionata o compressa), inoltre, è difficilmente applicabile ad alcuni formati compressi come il jpeg. Esistono altri metodi che non sono di tipo sostitutivo che garantiscono migliori prestazioni.

POSSIBILI ATTACCHI

Prima di esaminare gli algoritmi vorrei fornire un nuovo contributo teorico. È importante capire come sia possibile "attaccare" la steganografia. È un errore pensare che il messaggio segreto o possa essere intercettato e decifrato oppure possa andare a buon fine e non essere scoperto. Vi possono essere anche delle situazioni intermedie che comunque garantiscono dei risultati parziali sufficienti alla sicurezza della trasmissione del messaggio segreto. Approntando una classificazione dei possibili attacchi alla steganografia si distinguono cinque livelli:

1. Solo steganografico;
2. messaggio di copertura noto;
3. messaggio segreto noto a posteriori;
4. algoritmo steganografico conosciuto;
5. messaggio scelto.

Gli attacchi sono condotti da persone che intendono violare e decifrare la trasmissione segreta. Nel caso della steganografia tale figura è conosciuta come steganalista e la corrispondente materia di studio è la steganalisi. Nella peggiore delle ipotesi lo steganalista non verrà a conoscenza della presenza di una trasmissione segreta, il caso diametralmente opposto, che è il più favorevole possibile vede lo steganalista conoscere il messaggio segreto oltre che l'algoritmo usato. Ma esaminiamo i vari livelli. Il primo si ha quando si intuisce che all'interno di un messaggio è nascosto qualcosa di segreto, in tal caso si conosce solo il messaggio steganografato. Situazione più favorevole si ha qualora si conosca oltre al messaggio steganografato anche quello di copertura, così il confronto tra le due entità consentirà un'analisi tesa alla individuazione dell'algoritmo usato e del messaggio nascosto, compito molto difficile sulla base di queste informazioni. Se a posteriori si viene a conoscenza del messaggio segreto è possibile approntare uno studio più organico per l'individuazione dell'algoritmo e per la conseguente decifratura di altri messaggi sullo stesso canale trasmissivo. Va detto che anche questo compito è arduo anche se più favorevole rispetto al secondo tipo di attacco. Se si conosce l'algoritmo usato ci troviamo nel quarto caso che è sicuramente uno dei più fortunati, infatti, successivi elementi steganografati con lo stesso metodo potrebbero essere decifrati. Si tenga presente, ad ogni modo, che tale attacco non garantisce l'immediata individuazione del testo nascosto giacché, in molti algoritmi si fa riferimento a delle password che potrebbero essere spesso cambiate. Quindi, anche la conoscenza dell'algoritmo potrebbe non essere sufficiente al compito dello steganalista. L'ultimo caso si ha quando si è a conoscenza oltre che del messaggio segreto anche dell'algoritmo e quindi l'unione dei due casi precedenti; si tratta della situazione più rosea che comunque non garantisce la decifratura di ulteriori messaggi sullo stesso canale.

UN PRIMO APPROCCIO CON L'ALGORITMO DI A.BROWN:S-TOOLS

Un algoritmo molto usato nel campo della steganografia è stato sviluppato da *Andrew Brown*. Tale software reperibile in rete in versione shareware prende il nome di *S-Tools*. Si tratta di un algoritmo sostitutivo, appartenen-

te alla famiglia LSB. S-Tools è in grado di nascondere più messaggi nello stesso oggetto che può essere una immagine in formato *jpeg, gif* o *bmp* oppure un file sonoro *wav.* I passi dell'algoritmo sono i seguenti:

1. Compressione dei file da nascondere;
2. cifratura mediante il programma MD5 con password scelta dall'utente;
3. generazione casuale di una serie di numeri che individuano i byte dove inserire il messaggio nascosto.

I primi due stadi sono dei preliminari e consentono di pretrattare i dati da nascondere secondo metodi distinti dalla steganografia; essi vengono, infatti, compressi, in modo che occupino meno spazio e criptati, per avere un ulteriore grado di sicurezza. Il primo punto consente, tra l'altro, di avere un testo di copertura di più esigue dimensioni. Il terzo passo è il più importante nell'ambito che stiamo esaminando. Tale sezione si occupa di iniettare il messaggio nascosto nei bit meno significati di alcuni byte scelti casualmente. Da sottolineare che la sequenza random è generata in funzione di una password ed è quindi sempre differente. Per cifrare il testo non sarà sufficiente conoscere l'algoritmo ma bisognerà possedere la password. Un esempio chiarirà il funzionamento del programma sviluppato da A. Brown. Consideriamo come file di copertura un *wav*. È noto che un file di questo tipo si ottiene dalla discretizzazione del segnale analogico iniziale, attraverso la simultanea applicazione dei due procedimenti di campionamento e quantizzazione.
In Windows la quantizzazione è attuata con 8 o 16 bit, cosicché si può attingere rispettivamente a scale di 256 e 655536 valori. Cambiare il bit meno significativo di alcuni byte che operano la digitalizzazione (*quantizzazione*), soprat tutto nel secondo caso, produce modificazioni impercettibili anche alla più sensibile delle orecchie. Come si può comprendere si tratta di un procedimento del tutto analogo a quello adottato per le immagini. La particolarità dell'algoritmo è che i byte che verranno modificati nel loro ultimo bit sono scelti casualmente e dipendono dalla password. Tale relazione byte-password è l'unico elemento che rimane riservato per evidenti motivi di sicurezza.

UN SECONDO ALGORITMO: TEXTO

Un interessante algoritmo prodotto da Kevin Maher attua la steganografia su testi. Un messaggio segreto viene nascosto all'interno di una frase opportunamente generata. Essendo l'autore inglese la frase generata sarà in tale lingua; focalizziamo ad ogni modo l'interesse sul metodo. Va detto subito che le frasi prodotte, pur avendo un significato grammaticale, assumono un discutibile valore semantico, apparendo, il più delle volte, come periodi demenziali. Si ottiene un risultato simile ad un gioco in voga qualche anno fa nei paesi anglosassoni, di nome "mad libs" (improvvisazioni demenziali) che consisteva nel riempire frasi con spazi vuoti apponendo sostantivi o verbi e generando quindi periodi dal significato spesso comico. Maher ha approntato un dizionario di parole divise in categorie che sono: *oggetti, luoghi, verbi, avverbi* e *aggettivi*. Inoltre, ha generato delle frasi modello (*template*) che contengono spazi vuoti da riempire con le parole prima classificate. Poiché ogni categoria è costituita da 64 alternative si possono codificare sei bit; essendo gli spazi vuoti cinque, il testo segreto per ogni frase può essere costituito al più da 30 bit. Ogni parola contenuta nel dizionario occuperà una posizione che è associata ad una configurazione di sei bit. Sulla base di questo criterio si attuano le due fasi di codifica e decodifica. Il tutto funziona se il trasmittente e il ricevente usano lo stesso dizionario. Inoltre, è facile intuire come l'esigua quantità di parole sia una nota negativa che aiuta il compito dello steganalista. Ad ogni modo, ampliando il dizionario, ed il numero di template, si può pervenire ad un metodo dalle prestazioni considerevoli. Migliori performance si ottengono anche in questo caso associando tecniche crittografiche.

ALCUNI CENNI PRELIMINARI SU ALTRI ALGORITMI

Per completezza di trattazione vanno citati altri metodi steganografici. Tra quelli esaminati dalle ricerche su internet mi sono soffermato su due che, oltre ad essere particolarmente interessanti, hanno il "pregio" di essere stati ideati da italiani. Di questi farò una breve descrizione. La libreria *Psteg* è stata prodotta da *Roberto Fabbri*, essa implementa una tecnica con permutazioni pseudocasuali. La chiave usata può essere o inserita a run time o essere attinta da keyring di chiavi pubbliche di PGP. Sono previste operazioni preliminari come la compressione e la crittografia, inoltre, utilizzando la libreria, è facile produrre programmi a partire da algoritmi noti.
Interessante è anche l'idea di *Andrea Mazzoleni*

Alcune varianti di S-Tools

S-Tools è un pacchetto shareware che contiene più programmi. I più importanti tra questi programmi sono: ST-BMP che consente di iniettare più file in un unico file bmp, con tecnica RGB a 24 bit, o in un file gif a 256 colori; ST-Wav che occulta messaggi all'interno di un file sonoro; interessante, infine, è ST-BMP che per nascondere le informazioni fa uso dei settori non utilizzati dei floppy. Questa ultima tecnica è resa possibile dall'analisi della FAT del dischetto. Ovviamente in questo caso un uso diverso dalla sola lettura per il floppy provocherebbe una probabile perdita delle informazioni steganografate.

Permutazioni pseudocasuali

Gli algoritmi con permutazioni pseudocasuali appartengono alla famiglia di LSB. Si basano sulla modifica dei bit meno significativi di file immagine o sonori. La differenza con i metodi esaminati è nella scelta dei byte da modificare, che non viene fatta come una sequenza seppure random come nel caso di S-Tools, ma come la permutazione di un sottoinsieme di tali byte. Una permutazione di n oggetti è la nuova disposizione degli stessi secondo un nuovo ordine. Il numero totale di permutazioni aumenta in modo esponenziale all'aumentare del numero di oggetti. In particolare le permutazioni di n oggetti sono n! (*n* fattoriale, si ricorda che il *n!=n*(n-1)*(n-2)*..*1* e che il fattoriale di *0* è *1*). Scegliendo come successione di byte da modificare una permutazione del sottoinsieme prescelto si perde la caratteristica di ordine, cosicché il messaggio segreto viene "sparpagliato" tra i byte del messaggio di copertura. La distribuzione non ordinata dei bit modificati tra i byte disponibili fornisce un ulteriore elemento di sicurezza di prevenzione da eventuali attacchi.

Soluzioni

che ha prodotto il programma *Stego*, che al pari di *texto* "genera" i messaggi steganografati, non utilizzando quindi oggetti di copertura dove iniettare il messaggio segreto. Si distingue da texto per la tecnica generativa usata, in questo caso infatti, si tratta di "imitare" lo stile di scrittura di un testo preso come riferimento. Si riscontrano in tale metodo tecniche proprie della linguistica computazionale, anch'essa trattate tra queste pagine qualche tempo fa. Si estraggono dal testo di riferimento informazioni statistiche, come le occorrenze delle lettere e si produce il testo steganografato rispettando le stesse caratteristiche. Così però, si genera un testo che non ha un senso nemmeno a livello di parole, si parla infatti di pseudo parole. Una valida alternativa esamina come elemento di riferimento non le singole lettere ma parole o spezzoni di esse di lunghezza prefissata. In questo secondo caso si ottiene un testo corretto sintatticamente ma non semanticamente.
La generazione del testo steganografato a partire dal testo di riferimento e dalle informazioni statistiche dello stesso non è banale. Si segue una filosofia simile adottata nel caso della generazione del codice Hufman, tecnica utilizzata per la compressione, con la quale si associano sequenze di bit di lunghezza diversa a seconda della frequenza delle singole parole o spezzoni di esse. Il software prodotto da Mazzoleni è in grado di generare i testi anche senza alcun testo di riferimento, opzione molto comoda alcuni casi. Trovate il software, insieme, a tutti gli altri citati nell'articolo, nella sezione Soluzioni presente nel CD-Rom allegato

TECNICHE DI WATERMARK E FINGERPRINT

Per concludere esaminiamo alcune interessanti applicazioni della steganografia. Un uso significativo si ha nel campo dei diritti d'autore. Per preservare il copyright, sempre più autori appongono dei watermark alle proprie opere. Di cosa si tratta? Restringiamo l'osservazione al caso di opere intese come immagini digitali, ovviamente il discorso si può estendere anche ad altri tipi di produzione d'autore. Supponiamo che un artista produca un ray tracing o comunque un'opera che intende collocare sul mercato come file grafico, è naturale che le probabilità che l'opera sia appetibile si moltiplicano se essa è presente su un canale pubblico e molto frequentato, quale internet. Una tale politica rende l'opera maggiormente popolare ma inevitabilmente la sottopone al rischio di copia non autorizzata, con la conseguenza che prima o poi possa essere spacciata come originale. La presenza di un watermark garantisce circa la paternità dell'opera. Il ragionamento è semplice.
Vengono più volte iniettate, all'interno del ray tracing, dei loghi o il nome dell'autore secondo tecniche steganografiche, ad esempio con S-Tools. Così, si potrà in ogni momento riportare in chiaro tali loghi attribuendo la paternità dell'opera al legittimo proprietario. La presenza di numerosi loghi iniettati salvaguarda da eventuali ridimensionamenti e compressioni del file.
Il fingerprint è una variante del watermark utilizzata per controllare diverse copie di uno stesso prodotto. Tale strumento è utile quando si vuole che la copia di un qualcosa, supponiamo un software, rilasciata a pagamento o come copia di valutazione non vada in altre mani. Si vuole evitare cioè che una sola licenza venga distribuita, con violazione dei diritti di autore e delle norme di rilascio del software, a più persone, ossia che sia installata su più macchine. In questo caso si allestisce una tabella con la quale si associa ad ogni copia rilasciata un codice.
Così il codice identificherà in modo univoco l'acquirente o comunque il beneficiario del software. Il codice viene iniettato sempre con tecniche steganografiche in alcuni dati apparentemente innocui associati al software, o che comunque abbiano altri scopi, come ad esempio immagini. Se quindi si riscontrano installazioni con uguali fingerprint si deduce che alcune copie sono state illecitamente distribuite, inoltre si può stabilire chi abbia prodotto la copia. Attenzione poiché la tecnica abbinata all'uso di internet permette alle case produttrici di software di controllare eventuali copie pirata. Come? Semplicemente prevedendo una funzione che si innesca quando si è connessi a internet che invia in modo nascosto i fingerprint ai server della propria casa produttrice del software.

CONCLUSIONI

Sintetizzare la steganografia con soli due articoli è stato un compito molto arduo. Ho comunque la presunzione di avere presentato gli aspetti salienti della tecnica e di avere analizzato le sfumature di maggiore interesse. Risulta evidente come sia attuale nell'ambito della sicurezza informatica.
Quindi consiglio a chi si occupa di tale aspetto di approfondire l'argomento. Io intanto preparo qualcosa di nuovo per il prossimo appuntamento. Vi aspetto.

Fabio Grimaldi

Robotica: Hardware e Software.

Programmare un braccio meccanico

Elettronica e programmazione

In queste pagine vogliamo iniziare la costruzione di un braccio meccanico che sia in grado di manipolare oggetti di dimensioni e peso anche considerevoli.

Le applicazioni dell'elettronica e dell'informatica ci hanno fatto abituare ad un mondo sempre più interconnesso e talvolta ad una certa astrazione dei metodi di comunicazione che diventano sempre più 'tecnologici' e 'digitali'. Le applicazioni sorprendenti delle tecnologie orientate alla comunicazione, possono, talvolta, fare passare in secondo piano le applicazioni orientate al controllo di sistemi hardware, la gestione dei quali diventa, giustamente, sempre più lontana dall'utente finale.

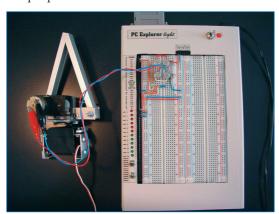



Fig. 1: Nella figura è visibile la realizzazione completa, composta dalla mano meccanica e dal circuito elettronico realizzato con l'apparecchiatura PC Explorer light: il lettore ha tutte le informazioni per realizzare questo progetto anche con le tecniche costruttive convenzionali.

Nell'intimo della parte 'hardwerista' del mio ego, in continua competizione con quella 'softwerista' rimane comunque il sogno di una apparecchiatura, che se vogliamo possiamo chiamare Robot, che ogni mattina mi porti la colazione a letto e che magari aiuti la mia consorte nei lavori di casa. Per fare contento mio figlio, magari potrebbe avere la forma di un robot dei più noti film, ma rigorosamente con capacità di manipolare oggetti di dimensioni e peso considerevoli. Tutto ciò è possibile e realizzabile da chiunque con un minimo di attrezzatura e seguendo quanto riportato in queste pagine e negli appuntamenti futuri. Per chi avrà voglia di seguirmi in questa affascinante avventura, potremo realizzare inizialmente un vero braccio meccanico che poi potrà essere inserito in un vero e proprio Robot, in grado di muoversi, di evitare ostacoli e di portare a termine compiti sempre più complessi. Le applicazioni hardware a questo punto potranno fondersi con le più moderne ed avanzate tecniche software, fino a portare ad una macchina sempre più versatile ed 'intelligente'.

IL BRACCIO MECCANICO

Il braccio meccanico che ci proponiamo di realizzare, una volta terminato sarà dotato di cinque gradi di libertà e potrà gestire la presa della mano, grazie a sensori di pressione: una serie di sensori infrarossi, inoltre saranno in grado di verificare se l'oggetto da manipolare è in posizione per la presa. Specificando meglio cosa intendiamo per cinque gradi di libertà, possiamo dire che il nostro braccio sarà dotato di capacità di presa della mano, rotazione del polso, movimento del polso, movimento del gomito e rotazione della spalla, in analogia con un braccio umano. La gestione elettronica del braccio avverrà per mezzo della porta parallela di un PC, per mezzo di specifiche di inter-

Fig. 2: Da una analisi dei dettagli costruttivi della mano meccanica, si nota che è completamente costruita in alluminio, sono visibili l'attuatore della presa utilizzato in questa sede ed il servomeccanismo deputato alla rotazione del polso che analizzeremo nei prossimi appuntamenti.

Filmati sul CD
\soft\media\braccio

File sul CD
\SOFT\CODICE\SPUNTO_Robot_Hand1.zip

File sul Web
www.ioprogrammo.net/files/72/Robot_Hand.zip

Bibliografia

• CONTROLLIAMO LA PORTA PARALLELA CON DELPHI 6
Luca Spuntoni
(ioProgrammo N57-58)
Aprile e Maggio 2002

Il braccio meccanico

Per maggiori informazioni sul braccio meccanico, sulle interfacce relative e sull'apparecchiatura 'PC Explorer light' è possibile visitare il sito:

http://web.tiscali.it/spuntosoft/

oppure scrivere all'indirizzo:

luca.spuntoni@edmaster.it

Elettronica e programmazione

Programmare *un braccio meccanico*

facciamento che verranno fornite di volta in volta. In questa sede analizzeremo come gestire la presa della mano meccanica: per ottenere maggiori dettagli costruttivi meccanici della realizzazione, che esulano dallo scopo di queste pagine, si consiglia di visitar il sito: *http://web.tiscali.it/spuntosoft/'* , oppure di contattare direttamente l'autore all'indirizzo di posta elettronica: *luca.spuntoni@edmaster.it*.

LA PRIMA REALIZZAZIONE: AZIONIAMO LA MANO MECCANICA

La mano meccanica, per essere in grado di manipolare oggetti di volume e peso consistenti deve essere azionata da un servocomando di potenza adeguata, nonché essere costruita in un materiale idoneo a maneggiare anche cose di composizione e caratteristiche fisiche diverse.
Per la nostra realizzazione è stata progettata una mano costituita da due dita, costruita in alluminio e di dimensioni paragonabili ad una mano umana, ovviamente fatte le dovute distinzioni.
L'attuatore elettromeccanico è un comune motore in corrente continua da *6 Volts*, dotato di motoriduttore *1:120*; deve essere detto che il rapporto di riduzione definisce in pratica, a parità di potenza del motore, la forza massima esercitabile con la mano, che ovviamente deve essere proporzionale al peso ed alle caratteristiche fisiche dell'oggetto da manipolare.
Il rapporto di riduzione definisce anche la velocità di apertura e di chiusura della mano, ma questo è un parametro secondario: le considerazioni di progetto importanti sono piuttosto orientate ad una corretta manipolazione degli oggetti, dal momento che la presa deve essere sufficiente a non fare cadere l'oggetto, ma non troppo forte da romperlo: penso infatti che mia moglie si sarebbe un poco alterata se le avessi frantumato la tazzina di porcellana di Fig. 3.

I componenti necessari

- 4 transistors 2N1711 o BC141
- 4 Resistenze da 1 K Ohm _ watt
- 1 motore CC per hobbistica.

Fig. 3: **La mano meccanica è stata progettata in modo tale che possa maneggiare oggetti di peso e dimensioni ragguardevoli: nella figura si nota il meccanismo in azione con una comune tazza da caffè.**

Per quanto riguarda l'hardware di gestione, consiglio come al solito di non utilizzare il computer nuovo di zecca appena acquistato.
L'utilizzo di un vecchio PC, magari dotato di Win95 è l'ideale per la sperimentazione elettronica e per la realizzazione di apparecchiature di controllo, in modo da non rischiare la 'vita' di macchine più nuove e pregiate: un PC 486 o Pentium può essere acquistato per poche decine di euro e garantisce ottime potenzialità dal punto di vista della sperimentazione.

IN FUTURO: I SENSORI DELLA PRESA E LA ROTAZIONE DEL POLSO

Nei prossimi appuntamenti termineremo la mano meccanica, conferendole la sensibilità tattile per mezzo di sensori di pressione posti sulle dita, posizioneremo e gestiremo inoltre dei sensori ad infrarossi per la verifica del corretto posizionamento dell'oggetto da manipolare.
Terminata la mano conferiremo al braccio il resto delle capacità di movimento: la rotazione del polso, il movimento del polso, il movimento del gomito e rotazione della spalla.
Terminato il braccio meccanico, potremo iniziare a sviluppare il resto del Robot, capace di muoversi agilmente, fungere da guardiano e maggiordomo di casa.

L'INTERFACCIA ELETTRONICA

Il problema che dobbiamo risolvere per rendere possibile il movimento della nostra mano meccanica è semplicemente quello di azionare il motore in corrente continua, dotato di motoriduttore rendendone possibile la rotazione nei due sensi (apertura e chiusura), nonché gestirne lo stato di arresto ed il fine corsa. Per quanto riguarda queste ultime caratteristiche, possiamo dire che esistono vari modi per implementarle e le possibili soluzioni verranno presentate nel prossimo appuntamento.
Per quanto riguarda il movimento del motore, conferiamo a due bit della porta Dati della porta parallela e per la precisione al bit D0 e D1, il compito di definire lo stato di Stop il Movimento di apertura ed il Movimento di chiusura.
Quando il livello logico sulla linea *D0* sarà *ALTO*, cioè *+5V*, il motore ruoterà in senso orario, causando l'apertura della mano, analogamente, quando il livello

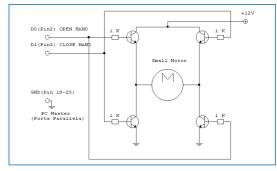



Fig. 4: **Lo schema elettrico del circuito necessario al controllo della mano meccanica è stato inserito, per comodità del lettore nel file incluso al CD (*SPUNTO_Robot_Hand.zip*) con il nome: *Schema_elettrico_Mano.bmp*.**

logico sulla linea *D1* sarà *ALTO*, il motore ruoterà in senso anti- orario, causandone la chiusura.
Durante le operazioni di apertura e chiusura l'altra linea dovrà essere a livello *BASSO*, cioè alla massa logica.
Gli stati corrispondenti ad entrambe le linee a livello *ALTO* od entrambe a livello *BASSO* causano l'arresto del motore, però di queste condizioni, soltanto quella relativa alle due linee a livello logico BASSO viene utilizzata come STOP del motore, per evitare l'inutile conduzione contemporanea dei quattro transistor.
Osservando lo schema elettrico, che per comodità del lettore viene incluso nel file contenuto nel CD allegato alla rivista, notiamo che quando entrambe le linee *D0* e *D1* hanno lo stesso stato logico, quindi *BASSO*, come abbiamo detto in precedenza, nessun transistor è in conduzione, dal momento che la differenza di potenziale tra ciascuna base ed emettitore è nulla, di conseguenza ad entrambi i capi del motore si trova una tensione nulla ed il motore è fermo.

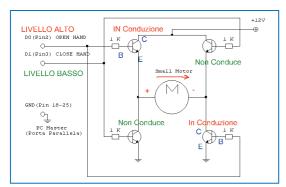



Fig. 5: Lo stato logico delle linee D0 e D1 determinano lo stato di apertura, chiusura o fermo della mano meccanica.

Quando una delle linee è a livello *ALTO*, sulla base dei transistor ad esso collegati, attraverso le apposite resistenze di carico, si troverà (senza scendere in dettagli di progetto che sarò felice di approfondire con i lettori interessati) una tensione sufficiente a portare i transistors in conduzione, annullando (o quasi) la differenza di potenziale tra i rispettivi collettori ed emettitori.
Questa situazione porterà a determinare una differenza di potenziale ai capi del motore e ne determinerà il senso di rotazione causando l'apertura o la chiusura della mano.
I transistor scelti nella applicazione (*2N 1711* o *BC141*) sono sufficienti per piccoli motori: è buona norma dotarli comunque di aletta di raffreddamento: nel caso di motori più grandi si consiglia di preferire transistor in configurazione *Darlington* oppure appositi circuiti integrati di potenza. Il dimensionamento dei circuiti in questo caso dovrà essere fatto in modo adeguato, per non assorbire troppa potenza dalla porta parallela del PC, con un possibile rischio di danneggiarla.

REALIZZAZIONE DEL CIRCUITO ELETTRICO

Per realizzare il circuito appena descritto a livello teorico, è consigliabile di procedere con ordine, iniziando ad eseguire per primi i cablaggi di tutte le linee di connessione, come indicato nella fotografia che segue.
Nell'immagine si notano, in alto a sinistra, le due connessioni alle linee della porta parallela *D0* e *D1*, nonché le resistenze di carico dei quattro transistor.
Si consiglia di eseguire i cablaggi con spezzoni di filo di colore appropriato, in modo da facilitare la ricerca degli errori.
Il cablaggio può essere eseguito facilmente utilizzando l'apparecchiatura mostrata in figura, chiamata *'PC Explorer light'*, la più semplice della famiglia *'PC Explorer'*, sulla quale è possibile avere maggiori informazioni sul sito *'http://web.tiscali.it/spuntosoft/'*.
Terminati i cablaggi, ai quali in modo opzionale si possono aggiungere i collegamenti ai due LED 1 e 2 come mostrato in figura, per monitorizzare lo stato

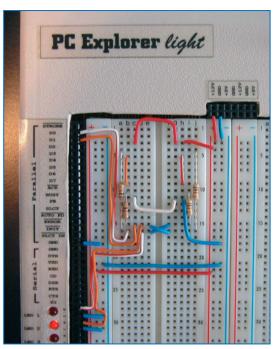



Fig. 6: Il cablaggio iniziale procede innanzi tutto posizionando le quattro resistenze e tutte le connessioni elettriche, come riportato nella foto di figura, maggiori informazioni sono reperibili sul il sito: *'http://web.tiscali.it/spuntosoft/'*.

PIN PC Master	SEGNALE	DIREZIONE	TIPO PORTA	PIN PORTA	DESCRIZIONE
2	D0	OUT	PARALLELA	2	Open Hand
3	D1	OUT	PARALLELA	3	Close Hand
18	GND	PARLLEL GND	PARALLELA	18-25	Signal Ground

Elettronica e programmazione

Programmare *un braccio meccanico*

Apparecchiature per montaggi sperimentali

Il sistema proposto in queste pagine è stato realizzato e collaudato con la apparecchiatura per il collaudo e la sperimentazione di circuiti elettronici con Personal Computer *'PC EXPLORER light'*: ulteriori informazioni su come si possa reperire questa apparecchiatura è possibile visitare il WEB all'indirizzo:

http://web.tiscali.it/spuntosoft/

oppure scrivere all'indirizzo: spuntosoft@tiscali.it

Elettronica e programmazione

Programmare
un braccio meccanico

delle linee *D0* e *D1*, possiamo procedere ad inserire i quattro transistors, facendo attenzione ad orientarli correttamente secondo quanto riportato in Fig.7.

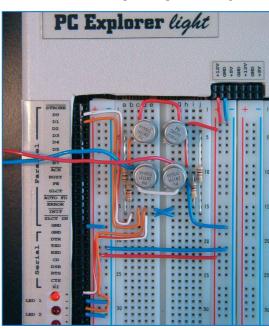



Fig. 7: **Terminato il cablaggio, siamo pronti ad inserire i quattro transistor, facendo attenzione che le tacche indicanti l'emettitore siano posizionate come in figura.**

A questo punto siamo pronti per collegare i terminali del motore della nostra mano meccanica, come mostrato qui di seguito dai fili rosso e blu che escono dalla breadboard: i fili possono essere invertiti se il verso di rotazione del motore, per un dato comando non è quello desiderato.
Occorre specificare che chi voglia costruire questa applicazione direttamente su circuito stampato, oppure con una piastra millefori, od ancora con una comune breadboard, può farlo seguendo lo schema elettrico riportato in questa sede e consultando le fotografie relative per verificare la correttezza dei cablaggi.

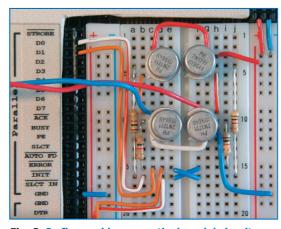



Fig. 8: **In figura si ha un particolare del circuito elettronico di controllo che è facilmente realizzabile con l'apparecchiatura PC Explorer light: è possibile cablare il circuito, in alternativa, utilizzando una comune piastra millefori ed i pochi componenti elencati a lato dell'articolo.**

Il software

Il software di controllo è stato collaudato con Win 3.x, Win 9x e Win Me, se si utilizza Win 2000, oppure NT, occorre scrivere una parte di codice che gestisce i privilegi del sistema, per non incorrere in un errore del tipo *'Privileged error'*.

IL SOFTWARE DI CONTROLLO

Il programma di controllo è scritto in Delphi, ma può essere tranquillamente trasportato in qualunque altro linguaggio con poche modifiche.
Nell'intestazione della unit principale notiamo che il programma utilizza in particolare *SpuntoLedComponent* e *UnitPortaParallela*, dei quali vengono forniti i files già compilati e pronti all'uso nel CD incluso alla rivista: di tutte le altre componenti del programma viene fornito il codice completo:

```
//****************************************//
//                                        //
// TSpuntoROBOTHand Delphi 6 Application Ver 1.0 //
//     Copyright 2003 Luca Spuntoni Jun. 2003    //
//     spuntosoft@tiscali.it              //
//                                        //
//****************************************//
unit SpuntoROBOT_Hand_Unit1;
interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
                    Graphics, Controls,Forms,Buttons,
  Dialogs, ExtCtrls, Menus, StdCtrls,
               SpuntoLedComponent, UnitPortaParallela,
  UnitAbout, Spin, Math, jpeg;
```

La classe principale, ingloba tutte le funzionalità principali del programma, in particolare notiamo l'oggetto *TTimer*, che ogni 0,5 secondi provvede a gestire il movimento della mano:

```
type
//*************** Main Class ***************//

 TSpuntoROBOTHandForm = class(TForm)
   StepperMotorPanel: TPanel;
   MainMenu1: TMainMenu;
   Files1: TMenuItem;
   Exit1: TMenuItem;
   ParallelManager1: TMenuItem;
   ShowManager1: TMenuItem;
   About1: TMenuItem;
   Label1: TLabel;
   Label2: TLabel;
   MoveHandCheckBox: TCheckBox;
   DirectionGroupBox: TGroupBox;
   OpenRadioButton: TRadioButton;
   CloseRadioButton: TRadioButton;
   ContinuousMonitoringCheckBox: TCheckBox;
   Image1: TImage;
   Image2: TImage;
   Image3: TImage;
   Timer1: TTimer;
   procedure ShowManager1Click(Sender: TObject);
   procedure Exit1Click(Sender: TObject);
   procedure About1Click(Sender: TObject);
```

```
procedure FormShow(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure HandAction;
procedure StopHand;
procedure CloseHand;
procedure OpenHand;
procedure DelaytoStepTimerTimer(Sender: TObject);
procedure ContinuousMonitoringCheckBoxClick(
                                    Sender: TObject);
procedure WritePort(PortAddress, PortData:word);
procedure MoveHandCheckBoxClick(Sender: TObject);
procedure OpenRadioButtonDblClick(Sender: TObject);
procedure CloseRadioButtonClick(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
 SpuntoROBOTHandForm: TSpuntoROBOTHandForm;
```

Fig. 9: All'esecuzione del programma si ha la schermata visibile in figura: i controlli disponibili permettono l'apertura e la chiusura della mano meccanica.

Nella parte *Implementation* della *Unit* analizziamo una alla volta tutte le procedure: dopo la creazione della form, viene lanciata l'esecuzione di *FormCreate*, che provvede, semplicemente, a predisporre i paramenti relativi allo stato iniziale delle checkbox della finestra:

```
implementation

{$R *.dfm}

procedure TSpuntoROBOTHandForm.FormCreate(
                                    Sender: TObject);
begin
 //Inizializzazione Stato dei componenti sulla Form
                                          principale
 MoveHandCheckBox.Checked:=False;
 ContinuousMonitoringCheckBox.Checked:=False;
 CloseRadioButton.Checked:=True;
end;
```

L'azionamento della mano viene controllata dalla procedura che segue, che verifica innanzi tutto se la mano debba essere mossa o meno e poi stabilisce in quale direzione, a questo punto chiama *OpenHand* per l'apertura e *CloseHand* per la chiusura.
Infine, se il movimento della mano deve essere arrestato, viene chiamato *StopHand*.

```
procedure TSpuntoROBOTHandForm.HandAction;
                          // Controls the ROBOT's Hand
Begin
    If MoveHandCheckBox.Checked then begin
      If CloseRadioButton.Checked then CloseHand
                                    else OpenHand;
    End
    Else Begin
      If SpuntoRobotHandForm.Visible then
        StopHand;
    End;
End;
```

Le procedure *OpenHand, CloseHand* e *StopHand* sono fondamentali, come è intuibile dal loro nome, per il movimento della mano: in pratica all'interno di queste procedure viene stabilito come debbano essere impostati i bit *D0* e *D1* della porta parallela, prima che venga chiamata la procedura Writeport, che scrive fisicamente il dato così ottenuto sulla porta.

```
procedure TSpuntoROBOTHandForm.StopHand;
                        // Stops the hand movement
Var
W,MyPortAddress:Word;
Begin
MYPortAddress:=FormPortaParallela.StepperMotorPorta
                    Parallela.SpuntoPortDataAddress;
    W:=0;          // Bit Do e D1 della porta=0
    WritePort(MyPortAddress,W);
End;

procedure TSpuntoROBOTHandForm.CloseHand;
                                 // Closes the hand
Var
W,MyPortAddress:Word;
Begin
MYPortAddress:=FormPortaParallela.StepperMotorPorta
                    Parallela.SpuntoPortDataAddress;
    W:=2;          // Bit D1 della porta=1
    WritePort(MyPortAddress,W);
End;

procedure TSpuntoROBOTHandForm.OpenHand;
                                  // Opens the Hand
Var
W,MyPortAddress:Word;
Begin
MYPortAddress:=FormPortaParallela.StepperMotorPorta
                    Parallela.SpuntoPortDataAddress;
    W:=1;           // Bit D0 della porta=1
```

Elettronica e programmazione

Programmare
un braccio meccanico

Open Hand e Stop Hand

Le procedure *OpenHand, CloseHand* e *StopHand* sono fondamentali, come è intuibile dal loro nome, per il movimento della mano: in pratica all'interno di queste procedure viene stabilito come debbano essere impostati i bit *D0* e *D1* della porta parallela, prima che venga chiamata la procedura Writeport, che scrive fisicamente il dato così ottenuto sulla porta.

Elettronica e programmazione

Programmare *un braccio meccanico*

```
    WritePort(MyPortAddress,W);
End;
```

La scrittura sulla porta parallela, e per la precisione sul byte *Dati* di questa periferica, avviene per mezzo del semplice codice assembler che viene riportato di seguito: il cardine è l'istruzione *OUT*, che permette la scrittura diretta su un indirizzo fisico di I/O della porta:

```
procedure TSpuntoROBOTHandForm.WritePort(
                                PortAddress, PortData:word);
// Write PortData over PortAddress if Port Writing
                                                is enabled
begin
   PortData := (PortData*256)+PortData;
   asm
   Mov ax,PortData
   Mov dx,PortAddress
   Out dx,ax
   end;
end;
```

Come dicevamo all'inizio di questo paragrafo, un timer innesca ogni 0,5 secondi il controllo della mano, questo viene fatto dalla procedura che segue, che provvede a richiamare appunto *HandAction*.

```
procedure TSpuntoROBOTHandForm.DelaytoStepTimerTimer(
                                          Sender: TObject);
begin
  HandAction;
end;
```

Le procedure che seguono sono tutte event handler dei comandi deputati al controllo della mano: semplicemente provvedono a chiamare *HandAction* ogni qualvolta viene fatto click per cambiarne lo stato:

```
procedure TSpuntoROBOTHandForm.MoveHandCheckBoxClick(
                                          Sender: TObject);
begin
   HandAction;
end;

procedure TSpuntoROBOTHandForm.OpenRadioButtonDblClick(
                                          Sender: TObject);
begin
   HandAction;
end;

procedure TSpuntoROBOTHandForm.CloseRadioButtonClick(
                                          Sender: TObject);
begin
   HandAction;
end;
procedure TSpuntoROBOTHandForm.Timer1Timer(
                                          Sender: TObject);
begin
   HandAction;
end;
```

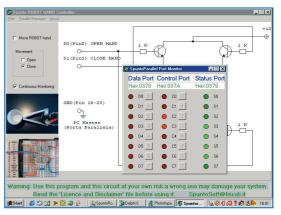



Fig. 10: **Attivando il 'Parallel Port Manager', con il comando relativo, è possibile visualizzare lo stato completo di tutte le linee della porta parallela: per maggiori informazioni consultare l'articolo: "Controlliamo la porta Parallela con Delphi 6", pubblicato su ioProgrammo N57-58 Aprile e Maggio 2002.**

Con questo programma è possibile monitorare, a livello di bit, lo stato dei tre byte di stato della porta parallela, leggendoli direttamente a livello hardware; questo può essere fatto lanciando le procedure che seguono, che provvedono a rendere visibile od a nascondere la form *FormPortaParallela*.

```
procedure TSpuntoROBOTHandForm.
  ContinuousMonitoringCheckBoxClick(Sender: TObject);
begin
FormPortaParallela.StepperMotorPortaParallela.
                        SpuntoContinuousMonitoring:=
                 ContinuousMonitoringCheckBox.Checked;
If  ContinuousMonitoringCheckBox.Checked Then
                          FormPortaparallela.Show else
                               FormPortaparallela.Hide;
end;

procedure TSpuntoROBOTHandForm.ShowManager1Click(
                                      Sender: TObject);
begin
 ContinuousMonitoringCheckBox.Checked:=True;
 FormPortaParallela.Show;
FormPortaParallela.StepperMotorPortaParallela.
                        SpuntoContinuousMonitoring:=
                 ContinuousMonitoringCheckBox.Checked;
end;
```

La form *About* viene resa visibile in modalità *modal* quando l'utente sceglie l'opzione *'About'*, oppure all'avvio del programma, prima della visualizzazione della finestra principale:

```
procedure TSpuntoROBOTHandForm.About1Click(
                                      Sender: TObject);
begin
SpuntoAbout.ShowModal;
```


L'autore

Luca Spuntoni si occupa da un decennio di interfacciamento di microprocessori e di sviluppo di applicazioni per l'ottimizzazione dei processi.

```
end;

procedure TSpuntoROBOTHandForm.FormShow(Sender:
                                        TObject);
begin
SpuntoAbout.ShowModal;
end;
```

Infine per terminare il programma viene eseguita la procedura che segue:

```
procedure TSpuntoROBOTHandForm.Exit1Click(Sender:
                                        TObject);
begin
SpuntoRobotHandForm.Close;
end;
```

Il programma Delphi, descritto in precedenza, ha la caratteristica di accedere all'hardware del PC attraverso i propri indirizzi fisici di I/O, questa tecnica, dal momento che scavalca il sistema operativo, potrebbe 'non piacere' a Windows NT, 2000, oppure XP, pertanto si consiglia di utilizzare un calcolatore dotato di Win 3.X, Win 9X, oppure Millennium.
In alternativa, occorre scrivere una parte di codice che gestisca i privilegi del sistema, per non incorrere ad un errore del tipo *'Privileged error'*.

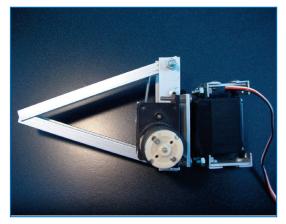



Fig. 11: **La mano meccanica è costituita da un azionatore dotato di un comune motore a corrente continua demoltiplicato in un rapporto *1:120*, maggiori informazioni ed una documentazione completa dei dettagli meccanici sono disponibili sul WEB all'indirizzo: *http://web.tiscali.it /spuntosoft/.***

COLLAUDO DEL SISTEMA

Innanzi tutto, prima di collegare il circuito al nostro PC, occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato collegato come previsto.
Colleghiamo al nostro circuito il cavo relativo alla porta parallela del PC, se possediamo PC Explorer come mostrato in figura, oppure provvedendo a costruirci un cavo seguendo lo schema elettrico e la tabella riportatinei paragrafi precedanti.

Fig. 12: **Dopo avere terminato l'assemblaggio del circuito, siamo pronti a collegare un comune cavo parallelo alla nostra apparecchiatura, oppure nel caso in cui il circuito sia stato autocostruito siamo pronti a collegarlo alla porta parallela del PC.**

Accendiamo il PC, lanciamo il programma di gestione e forniamo alimentazione al circuito: a questo punto cliccando sui comandi di apertura e chiusura dovremmo vedere la nostra mano muoversi. Nel caso in cui l'azionatore si muova in direzione opposta al previsto, provvediamo semplicemente a scambiare le connessioni del motore. Il circuito funziona anche per comandare un comune motore in corrente continua per altre applicazioni: si raccomanda di dimensionare i transistor in modo adeguato, per non sovraccaricare la porta parallela, cosa che comporterebbe qualche rischio di danneggiamento del PC.
Se il circuito non funziona, provvediamo a spegnere tutto prima di ricontrollare i collegamenti e riprovare di nuovo.

CONCLUSIONI

In queste pagine abbiamo visto come controllare il movimento di una mano meccanica azionata da un motore in corrente continua: il progetto dello schema elettrico, tutti i collegamenti necessari, il software compilato ed i relativi codici sorgenti sono stati messi a completa disposizione del lettore.
Il lettore vorrà comprendere che nonostante quanto esposto in queste pagine sia stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo e di studio, pertanto l'editore e l'autore non sono da considerare responsabili per eventuali conseguenze derivanti dell'utilizzo di quanto esposto in questa sede, soprattutto per la tipologia e la complessità dell'argomento.
Per maggiori informazioni sul braccio meccanico, sulle interfacce relative e sull'apparecchiatura *'PC Explorer light'* è possibile visitare il sito :*'http://web.tiscali.it/spuntosoft/'*, inoltre l'autore è lieto di rispondere ad ogni richiesta di chiarimento o delucidazione sull'argomento all'indirizzo di posta elettronica *luca.spuntoni@edmaster.it*.

Luca Spuntoni

Elettronica e programmazione

Programmare
un braccio meccanico

Precauzioni

Prima di collegare il circuito al nostro PC occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato collegato come previsto.

Servizi multimediali con Flash Mx e Web Service.

Flash

Previsioni meteo animate

Macromedia Flash Mx e i Web Service alfieri della nuova programmazione orientata a Internet si candidano prepotentemente a guidare questa rivoluzione. Scopriamo la potenza della loro unione.

File sul CD
\soft\codice\Meteo.zip

File sul Web
www.ioprogrammo.net/files/72/Meteo.zip

Macromedia Flash Remoting

Macromedia Flash Remoting consente agli sviluppatori ColdFusion di rendere facilmente disponibili servizi remoti ai client Macromedia Flash. Utilizzando ColdFusion Components o il nuovo supporto per ActionScript sul versante server, gli sviluppatori che hanno familiarità con ColdFusion o Macromedia Flash possono facilmente creare applicazioni che uniscono la sensibilità e la funzionalità di applicazioni client/server con i bassi costi di sviluppo consentiti da Internet. Incluso come servizio nativo in ColdFusion MX, Macromedia Flash Remoting supporta anche le connessioni dirette con i client Macromedia Flash Player sia verso i componenti .NET, che J2EE.

I vantaggi dei Web Service sono evidenti: sono indipendenti dalla piattaforma, sono trasparenti per i firewall e sono versatili.Già adesso stiamo assistendo a una convergenza delle tecnologie dei portali con le architetture basate sui Web Service. Complesse architetture possono essere suddivise in piccoli oggetti modulari, che possono essere condivisi e riutilizzati da utenti autorizzati o da agenti intelligenti. Queste caratteristiche permettono di creare framework come se si stesse disegnando un sistema OOP. Come risultato, ad esempio, un'azienda potrebbe introdurre un nuovo prodotto senza dover riconfigurare la tecnologia esistente, in quanto i moduli di processo basati sui servizi verrebbero riutilizzati per aggiungerlo all'infrastruttura. Riutilizzando servizi già pronti, il costo e il tempo necessari a integrarsi con nuovi sistemi, o compiere dei cambiamenti su quelli esistenti, dovrebbe ridursi significativamente. Flash MX è comodo perché permette di creare animazioni usabili ed ergonomiche, caratterizzate da layout accattivanti e d'impatto grazie al rendering di grafica vettoriale. Chi è abituato a sviluppare codice HTML è alle prese con problemi di compatibilità fra i vari browser: Flash MX garantisce invece un ottimo supporto cross-browser. Ciò significa che una volta che il movie sarà completato, esso sarà visibile allo stesso modo su qualsiasi client e su ogni piattaforma. Ma tutto questo è il meno: c'è molto di più! La grande cosa che ci permette Flash MX è di usare Remoting MX. La straordinarietà di Remoting è che ci fa invocare dei metodi direttamente sul server, catturandone il risultato senza nessun parsing particolare da parte nostra. In pratica, possiamo comunicare direttamente con ColdFusion MX (che usiamo nel nostro esempio) o con server Java o .NET. Il tutto soltanto con qualche riga di Actionscript, il linguaggio interno di Flash, "cugino" di ECMA script, padre dei moderni linguaggi di script. Per rendere l'idea, considerate di avere un metodo sul server (scritto quindi in Java o .NET) che si chiama getName() che restituisce una stringa dopo una connessione al database. Con Flash MX, tramite Actionscript lo invocate semplicemente con oggetto*Service.getName()*, ottenendo il risultato in una funzione di callback che si chiama *getName_result*.

MIRACOLI DELLA SERIALIZZAZIONE!

Ciò apre le porte a un ordinato e vantaggioso uso della programmazione OOP tramite Actionscript applicando così alle nostre architetture anche i più utili design pattern. Il tutto diminuendo la banda utilizzata poiché il Remoting MX prevede una comunicazione client-server secondo il formato AMF (Action Message Format), che è binario e compresso. Niente male per uno strumento praticamente utilizzabile dalla stragrande maggioranza dei client. Web Service e Flash MX vanno a braccetto permettendo di realizzare una perfetta scalabilità, e rendendo così l'azienda agile e flessibile nei cambiamenti. Entrambi usano il protocollo HTTP raggiungibile sulla porta 80 e sono modellati su soap. Lavorando su progetti di dimensione almeno media, spesso si incontrano notevoli difficoltà e problemi legati alla sicurezza, quando fra client e server si frappongono firewall e proxy. È un problema che in genere affligge altri standard come CORBA, DCOM e RMI. Fortunatamente a noi va bene perché la porta 80 risulta sempre aperta. Vediamo ora quali passi dobbiamo seguire per invocare un Web Service remoto e manipolarne la risposta da flash. Realizziamo un movie che ci informa sul meteo in tutto il mondo.

PREPARIAMO L'AMBIENTE

Come detto, Flash Remoting può comunicare con il server in modo diretto tramite Remoting. Sul server andremo ad installare ColdFusion MX (presente nel CD allegato al numero 71 di ioProgrammo e scaricabile dal sito ufficiale di Macromedia in prova all'indirizzo *http://www.macromedia.com/software/coldfusion/trial/*) che è di immediata installazione, non richiede

nessuna configurazione, basta solo lanciare il setup, leggere le informazioni che si presentano e scegliere avanti, avanti… fine. Sia ben chiaro che non scriveremo nessuna riga di codice lato server, esso fungerà soltanto come una specie di proxy. Il servizio di Flash Remoting non deve essere configurato poiché è installato automaticamente con ColdFusion MX. Una volta sistemato ColdFusion, vanno installati lato client i Flash MX Remoting Components , un piccolo pacchetto scaricabile dal sito macromedia all'indirizzo *http://www.macromedia.com/software/flashremoting/downloads/components/* che si integra con l'ambiente Flash MX. In particolare fornisce dei file che vanno importati nell'actionscript, essi sono:

- **"NetServices.as"** che espone una serie di oggetti e metodi che permettono di usare Remoting.
- **"NetDebug.as"** che permette di utilizzare un valido strumento di debug.
- **"DataGlue.as"** che binda un dataprovider (ad es un recordset) a un oggetto che lo visualizza (es combobox).

DESCRIZIONE DEL NOSTRO WEB SERVICE

Vi sono molti web service disponibili gratuitamente in rete, noi andremo a invocarne uno che offre dei report sullo stato meteorologico in molte città del mondo:

http://www.webservicex.net/globalweather.asmx?WSDL

Per le vostre prove dovrete quindi essere connessi in rete. Esso, oltre ad essere gratuito, ben si presta ad essere analizzato in quanto dà un tipo di risposta molto utile a fini didattici. Esso espone questi metodi:

GetCitiesByCountry(), che restituisce tutte le città principali di un paese specificato, rispondendo in questo modo:

```
<NewDataSet>
  <Table>
    <Country>paese scelto</Country>
    <City>citta1</City>
  </Table>
  <Table>
    <Country> paese scelto </Country>
    <City>citta2</City>
  </Table>
…..
</NewDataSet>
```

GetWeather(), che ci dà il report sullo stato meteo della città specificata e risponde in questo modo:

```
<?xml version="1.0" encoding="utf-16"?>
<CurrentWeather>
  <Location>città</Location>
  <Time>informazioni orarie</Time>
  <Wind> direzione e velocità del vento</Wind>
  <Visibility> distanza visibilità</Visibility>
  <Temperature> dati temperatura </Temperature>
  < SkyConditions> condizione del cielo</ SkyConditions >
  <RelativeHumidity> percentuale umidità
                                        </RelativeHumidity>
  <Pressure> pressione atmosferica</Pressure>
  <Status>esito operazione</Status>
</CurrentWeather>
```

Come risposta avremo solo i nodi disponibili, per cui alcuni potrebbero non essere ricevuti.

COSTRUIAMO IL MOVIE FLASH

Il movie flash che andiamo a costruire contiene solo qualche componente e poche righe di codice Actionscript. Lanciate Flash MX e create un nuovo movie. Inserite nello stage due combobox. Alla prima assegnate il nome di istanza *countries_cb* e alla seconda *city_cb*. Aggiungete un *PushButton*, chiamatelo *report_btn* e come *click handler* settate *callWeatherService*, la funzione *Actionscript* che chiederà il report. Selezionate *"countries_cb"* e, dal pannello delle properties, valorizzate label e data della combobox, inserendo i nomi degli stati che vi interessano. Dal pannello, inoltre, settate il *change handler* della combo a *"getCity"* o a una di vostra scelta. Tramite questo handler possiamo eseguire una funzione personale quando selezioniamo una nuova voce: nel nostro caso, ogni volta che scegliamo uno stato diverso, dobbiamo chiedere al Web Service di fornirci la lista di città di cui dispone di report. La lista ricevuta verrà caricata sulla combo *city_cb* in modo dinamico, tramite Actionscript, settando un *dataProvider* oppure eseguendo degli *addItem*. Lascio a voi la gioia di inserire qualche bello sfondo, o effetto animato che più vi aggrada, per rendere gradevole il movie. A questo punto, per maggior ordine aggiungete un layer e chiamatelo *"acts"*. Nel primo frame di questo livello andrà inserito il codice Actionscript. In Fig. 1 potete osservare una schermata su come organizzare il movie. Dopo aver incluso i file che ci permettono di usare *Remoting* (è necessario solo *NetServices.as* una volta che il movie viene compilato per il server di regime), per poter chiamare un web service o una procedura remota in Actionscript, basta seguire questi passi:

1) Si specifica e ci si connette al gateway, ottenendo un oggetto *NetConnection:*

```
NetServices.setDefaultGatewayUrl(
    "http://localhost:8500/flashservices/gateway");
this.gateway_conn =
            NetServices.createGatewayConnection();
```

Flash

Previsioni
meteo animate

ColdFusion MX e Web Service

"Le funzionalità Web service e XML di ColdFusion MX sono estremamente potenti, permettendo di realizzare in tempi brevi ciò che altre tecnologie richiederebbero naggiore tempo di sviluppo.
Utilizzando le nuove e rivoluzionarie funzionalità di ColdFusion Components integrate in ColdFusion MX, gli sviluppatori possono facilmente incapsulare e riutilizzare il codice per creare applicazioni ben strutturate che possono essere consultate automaticamente come Web service o servizi remoti per i client Macromedia Flash che utilizzano il servizio integrato Macromedia Flash Remoting.

Previsioni
meteo animate

Fig. 1: Come organizzare il movie.

Notare che la sintassi del gateway è questa:

```
http://[servername]/flashservices/gateway
```

Sia chiaro che *flashservices/gateway* non è un percorso realmente esistente sul server, è solo una mappatura logica per il servizio Flash Remoting in ColdFusion MX. Avviene tutto in modo trasparente, senza configurare nulla, senza scrivere alcuna riga di codice CFML!

2) si crea un riferimento al servizio sul *NetConnection* (creo *service object*), es:

```
this.myService = gateway_conn.getService(
        "http://www.webservicex.net
        /globalweather.asmx?WSDL", this);
```

3) si invoca il servizio sull'oggetto *service* specificando l'oggetto di richiesta, es:

```
this.myService.GetWeather({CityName:"Milano /
        Malpensa",CountryName:"Italy"})
```

Questa istruzione provvede a invocare il metodo remoto implementato sul server e che ci espone il web service. Per ottenere la risposta abbiamo principalmente due alternative:

- settare un oggetto ricevente sulla *getService*, in questo caso nell'oggetto ricevente bisogna implementare un metodo il cui nome è così composto:

```
nomefunzioneserver_result(result)
```

mentre per informazioni di stato va usato *nomefunzioneserver_status* (stato).

oppure

- se il ricevente non è settato nella *getService*, esso va settato sulla funzione chiamante come primo parametro creando un *new OggettoRicevente()*, oggetto che al suo interno implementa i metodi *onResult* e *onStatus*, es:

```
function OggettoRicevente () {
   this.onResult = function(result)    {
        //dati ricevuti
        trace("Dati ricevuti : " + result);
   }
   this.onStatus = function(error) {
        // controllo errore
        trace("Errore : " + error.description);
   }
}
```

Nell'esempio che stiamo costruendo invochiamo i metodi *GetCitiesByCountry* e *GetWeather*, andiamo quindi a definire due callback che si chiamano rispettivamente *GetCitiesByCountry_result(result)* e *GetWeather_result(result)*. Entrambe vengono invocate in modo automatico quando arriva la risposta del server. Questi metodi ricevono un argomento che si chiama result che è un oggetto che contiene la risposta fornita dal web service. Nel nostro esempio riceviamo un oggetto *xml*, per usarlo da flash bastano poche righe di codice, per es:

```
var cities_xml= new XML();
cities_xml.ignoreWhite=true;
cities_xml.parseXML(result);
```

In pratica istanziamo un nuovo oggetto, forziamo che i nodi di testo che contengono solo spazi bianchi vengano eliminati durante il processo di analisi sintattica *(ignoreWhite=true)* ed eseguiamo l'analisi del testo XML ricevuto, inserendo gli elementi dell'albero nell'oggetto *(parseXML)*.
Tramite il DOM *(document object model)* possiamo estrapolare le informazioni che ci servono andando a scorrere i nodi *child*. Quando riceviamo la lista di città popoliamo la combo, quando riceviamo il report meteo organizziamo i dati in un hash secondo questo generico schema:

```
hash [nodes[i].nodeName]= nodes[i].firstChild;
```

Se avete bisogno di fare debug basta che all'interno di Flash Mx scegliate *Window/NetConnection Debugger*. Esso è un validissimo quanto utile strumento che permette di vedere cosa stia accadendo dietro le quinte. Vengono visualizzati i dati ricevuti insieme a una specie di log delle chiamate.

Distribuire su .NET o su J2EE

Flash Remoting MX per Microsoft .NET è una implementazione pura di .NET eseguita come codice gestito al 100%, che sfrutta i vantaggi offerti da .NET, garantendo prestazioni ottimali, funzionalità e sicurezza. Flash Remoting MX per Java è una soluzione Java pura al 100%, che può essere distribuita come file WAR o EAR su server applicativi Java conformi.

Se qualcosa non va verrete avvisati subito. Un errore banale ma tipico consiste nel chiamare i metodi remoti passando parametri con tipo diverso. Per esempio se dovete passare una stringa al metodo remoto accertatevi che sia effettivamente una stringa anteponendo “”+ variabile oppure facendo *new String (variabile)*.
L’errore è banale e tipico perché Flash non è rigoroso sui tipi di dato, mentre il metodo remoto scritto sul server per es. in java o .NET sì. In Fig. 2 potete vedere un esempio del debugger.

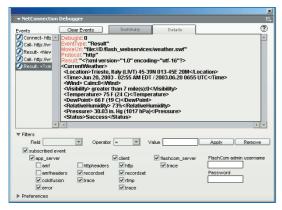



Fig. 2: **NetConnection Debugger in azione.**

Costruiamoci ora un oggetto con le informazioni che ci servono attingendo dall’hash i valori:

```
var reportData=
{
    // costruisco oggetto con i dati di report
    Location:hash.Location,
    Wind: hash.Wind,
    Visibility: hash.Visibility,
    Sky: hash.SkyConditions,
    Temperature:hash.Temperature,
    Pressure: hash.Pressure
}
```

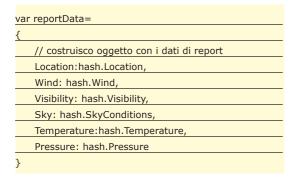
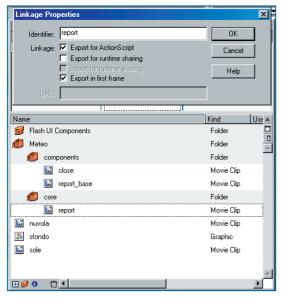



Fig. 3: **Come settare il linkage al componente.**

Esso ci tornerà utile per rappresentare in modo agevole i dati raccolti. Per visualizzare il report, ci creiamo infatti un componente Flash. Aprite la libreria *(CTRL-L)* e aggiungete un nuovo simbolo, chiamandolo *“report”* assegnando come itentifier di linkage sempre *“report”* come in Fig. 3.
Editiamo il componente e inseriamo dei campi di testo con opportuni nomi di istanza che andremo a valorizzare tramite actionscript così:

```
this.nome_istanza.text=
                    this.informazione_da_rappresentare;
```

Vi chiederete da dove salterà fuori *“informazione_da_rappresentare”*, e qui viene il bello, e viene gratis. Infatti *“informazione_da_rappresentare”* non è altro che una generica proprietà dell’oggetto *reportData* che abbiamo creato, e che viene passato al componente in fase di creazione tramite questa istruzione:

```
this.attachMovie ("report","report_mc",10, reportData)
```

Questa istruzione permette di prendere da codice un elemento dalla libreria e di metterlo nello stage. Deve avere specificato l’itentifier di linkage, un nome di istanza e un livello su cui posizionarsi. Il quarto parametro è il nostro oggetto personale costruito in precedenza.Se avete fatto tutto bene dovreste visualizzare una scheda come quella in Fig. 4.

Fig. 4: **Fotoapp.bmp, didascalia=esempio di applicazione.**

Se avete dubbi o qualcosa non vi è riuscita, allegato alla rivista c’è il movie d’esempio.

CONCLUSIONI

Abbiamo costruito lo scheletro minimo che permette di invocare un Web Service da Flash MX e senza scrivere nessuna riga di codice sul server!
A voi non resta che migliorare e approfondire questa struttura base inserendo meccanismi di caching e controlli sulla correttezza ed esistenza dei dati ricevuti.
Tenete presente che sarà molto semplice creare un Web Service personale, qualora non disponiate di un servizio gratuito già pronto; ciò sarà oggetto di altri articoli.

Marco Casario

Distribuire su qualsiasi periferica

Flash Remoting permette di distribuiee in modo uniforme, ad oltre 455 milioni di utenti Internet, attraverso le principali piattaforme e periferiche. Macromedia Flash Player è l'applicazione rich-client più affermata, installata su oltre il 98% di tutte le periferiche abilitate ad Internet. La distribuzione di applicazioni Internet dinamiche create con Macromedia Flash MX e Flash Remoting MX non richiede la distribuzione di software client addizionale.

Applicazioni ben concepite

Con Flash Remoting, è possibile creare applicazioni Internet dinamiche ben concepite per implementare un modello a tre livelli oppure a n livelli, con una netta separazione tra logica di presentazione e logica aziendale e dati. Inoltre, utilizzando le funzioni interne del server applicativo, quali la sicurezza e la gestione delle sessioni, Flash Remoting MX garantisce l'integrazione dei meccanismi di sicurezza e di gestione delle sessioni tra i client Macromedia Flash ed i server applicativi.

Gli strumenti degli hacker.

Buffer OverFlow: l'arma segreta degli hacker

Sicurezza

"Esiste un sottile confine fra l'esser d'aiuto agli amministratori per proteggere i loro sistemi ed il fornire strumenti utili agli hackers."
[R. Morris, "UNIX Operating System Security"]

File sul CD
\soft\codice\CodiceBO.zip

File sul Web
www.ioprogrammo.net/files/72/CodiceBO.zip

Java è più sicuro
È importante notare che C/C++ è il linguaggio più vulnerabile agli errori di overflow a causa delle sue potenzialità; esso infatti consente di accedere e referenziare la memoria in maniera diretta. Java evita questa classe di vulnerabilità perché ha una gestione molto sofisticata della memoria, ricorrendo all'uso di algoritmi di de-allocazione e a sistemi come il Garbage Collector.

Tre novembre 1988, una data che per molti non significa nulla, ma che per una ristretta cerchia di persone ricorda invece uno dei giorni più neri della storia dei computer, quando una delle prime versioni della rete Internet, veniva bloccata e congestionata da un'entità anomala a cui sarebbe stato dato il nome di *"worm"*. I più preparati in materia ricorderanno senz'altro questa triste vicenda, causata quasi per scherzo da uno studente di nome R. Morris e dal suo codice malevolo, scritto in C, capace di infettare a catena le workstation con sistemi SunOS e BSD Unix, ospitate dal MIT e da Berkeley. Parte del codice infettivo di questo worm si basava su un tipo di attacco nuovo nel suo genere, che interessava il servizio finger di Unix e che implementava il primo esempio di buffer overflow conosciuto della storia. Anno 2003, circa sette anni più tardi. La rete Internet è cresciuta a dismisura e ricorda lontanamente il suo predecessore; il World-Wide-Web è diffuso ovunque e costantemente vengono creati nuovi servizi di rete, mentre le tecnologie di routing e di trasmissione hanno fatto passi da gigante; i sistemi operativi e i linguaggi si sono evoluti notevolmente…ma, strano a dirsi, tutto ciò non basta per fermare l'avanzata del worm *SQL-Slapper*, che il 25 Gennaio 2003, proprio come avveniva alcuni anni prima, riusciva a paralizzare la rete mondiale per diverse ore. Anche questa volta il problema, legato questa volta ad un prodotto Microsoft, dipende da un *buffer overflow*.

Fig. 1: Internet Health è un indice di misurazione dello stato di Internet a livello mondiale.

GLI OVERFLOW

Come si è capito da questo breve preambolo, il *"Buffer Overflow"* è diventato, col tempo, una delle forme di aggressione più diffuse e potenzialmente dannose contro i sistemi informatici. Nato originariamente in ambiente Unix, proprio perché tale sistema era concepito interamente in linguaggio C, oggi il buffer overflow è un problema che interessa, senza esclusioni di sorta, sistemi operativi di ogni tipo (Unix, Linux, Win32, SunOS) e diverse architetture di calcolo (Intel, Alpha e MIP). Per rendersi conto della gravità del problema e dell'estensione del fenomeno basta dare un'occhiata alla mailing list Bugtraq (*http://www.securityfocus.com/archive*) o alla top-20 dell'istituto SANS (*http://www.sans.org/top20/*), che pubblicano ogni giorno le vulnerabilità più diffuse su Internet: quasi tutte sono dovute ad errori di buffer overflow. In apparenza, quando si verifica una condizione di overflow all'interno di un programma, a prima vista il programmatore sembra avere a che fare con un errore di poco conto, che nel peggiore dei casi può causare la terminazione imprevista di un'applicazione, lasciando un noioso messaggio di "crash" del programma. Tuttavia la condizione di overflow, come si è scoperto nel corso degli anni, è un problema ben più grave perché, se sfruttata pienamente, e in modo efficace da un hacker, può consentire ad un aggressore di iniettare e far eseguire, anche da remoto, qualsiasi tipo di codice "malizioso" sul sistema vittima, compromettendone la sicurezza. Oggi si parla spesso di attacchi che sfruttano la condizione di buffer overflow, ma sono veramente poche le persone che capiscono a fondo e conoscono i meccanismi di queste vulnerabilità; altri invece hanno solo una vaga idea di cosa siano, mentre altri ancora

Sicurezza

Buffer OverFlow: l'arma segreta degli hacker

ritengono queste forme di attacco una specie di "segreto mistico" in mano ai guru e agli hacker della programmazione. Al contrario non si tratta di nulla di così eccezionale e misterioso come sembra, il buffer overflow – come vedremo a breve - è soltanto un problema legato alla cattiva scrittura del codice da parte dei programmatori e alla mancanza di adeguati controlli nei sorgenti, null'altro. In questo articolo, oggi, cercheremo di mostrare i principi che stanno alla base degli overflow, illustrando anche alcuni concetti (come lo *stack* e le *function calls*) che ci aiuteranno a capire meglio il codice presentato e serviranno per sperimentare da vicino qualche overflow elementare. Le conoscenze richieste per una corretta lettura e per la piena comprensione degli argomenti trattati sono le basi del linguaggio C/C++ e i rudimenti di Assembly, con particolare riguardo per il funzionamento dei registri, dello stack, delle chiamate e del controllo di esecuzione all'interno di un programma. Nel presente articolo ci limiteremo a trattare una categoria molto comune di buffer overflow, che va col come di *"stack-based overflow"*; è comunque utile ricordare che oggi esiste una varietà di overfllow (*heap overflow, frame pointer, adjacent memory*, ecc.) più o meno complessi da implementare, la cui trattazione esula da queste pagine. Tutti i sorgenti considerati sono concepiti per lavorare sotto Win32, ma con poche modifiche sono facilmente portabili anche sotto Linux.

LO STACK: IL CUORE DEL PROBLEMA

Dovendo dare una definizione più formale del problema descritto, potremmo dire che l'overflow è quel tipo di errore che occorre in un programma, al tempo di esecuzione, quando una certa istruzione tenta di scrivere dentro ad un buffer più informazioni di quante esso possa contenerne. Un buffer può essere visto come una sorta di contenitore di dati, anche se a livello fisico si tratta semplicemente di una preservazione di una certa zona della memoria (*allocazione*) mediante l'assegnazione di un indirizzo (*offset*) di partenza e di fine, che segnalano ad un programma dove risiede l'area di memoria dedicata al contenimento di determinate informazioni. L'overflow avviene quando i dati scritti nel buffer oltrepassano i limiti prestabiliti, andando a modificare altre variabili adiacenti in memoria o (nella peggiore delle ipotesi) andando ad alterare il codice che è in fase di esecuzione, cosa a cui aspirano tutti gli hacker. Come si vedrà a breve, il pericolo maggiore dell'overflow sta proprio nella possibilità di prendere il controllo di un programma, al tempo di runtime, modificando il flusso normale delle istruzioni e dirottandolo (*hijack*) su altre istruzioni inserite dall'esterno nella memoria. Come si evince dalla Fig. 2, un programma in fase di runtime è mappato in memoria attraverso alcune strutture logiche. Il segmento codice viene generalmente caricato in testa alla memoria e contiene le istruzioni, in linguaggio macchina, da eseguire sulla CPU; segue quindi l'area dati, di dimensione predefinita, che contiene le variabili "statiche" del programma (vettori di dimensione fissata, stringhe di testo, interi). La parte restante di memoria viene invece gestita dinamicamente al momento dell'esecuzione e si divide (solo a livello logico, perché non esiste in realtà una divisione netta) in *Heap* e *Stack*; lo stack è quell'area che si trova sul fondo della memoria e che viene utilizzata durante le chiamate alle funzioni per salvare lo stato corrente dell'esecuzione e per passare i parametri per argomento; l'heap è invece in cima ed è usato per gestire variabili dinamiche e puntatori (ad esempio quando in C++ si richiama l'operatore *new* o l'istruzione *malloc*).

Fig. 2: Schema logico di come è mappato un file eseguibile al momento del runtime.

GESTIONE DELLO STACK

Lo stack viene gestito con politica LIFO *(Last In, First Out)* e mediante due istruzioni fondamentali del linguaggio Assembly: *PUSH* e *POP*. La prima memorizza un dato, la seconda lo estrae seguendo il modello *LIFO*, l'ultimo ad entrare è il primo ad uscire. La memorizzazione dei dati nello stack segue un ordine inverso, cioè parte dal fondo e man mano che le informazioni vengono memorizzate, sale verso la cima.
La gestione dello stack è affidata a due registri a 32-bit molto importanti, chiamati *EBP* e *ESP* (*base pointer* e *stack pointer*), usati – rispettivamente – per delimitare la cima e il fondo dello stack; quando si esegue una *PUSH*, il registro *ESP* viene decrementato di un numero di byte pari alla dimensione del dato memorizzato, quando si esegue una *POP* il registro *ESP* viene invece incrementato. Le variazioni di *ESP* e *EBP* possono inoltre essere fatte anche direttamente, tramite operazioni di somma e sottrazione (*ADD* e *SUB*): ad esempio l'istruzione *SUB ESP,5* riserva cinque byte nello stack. Questo tipo di struttura si rivela efficiente e utile nella gestione delle chiamate di funzioni e procedure in un programma, costituendo quello che è il meccanismo della *"call chain"*: una funzione può infatti richiamare un'altra funzione che a sua volta può chiamarne un'altra ancora e così via. Il sistema operativo deve poter tenere traccia di tutte le chiamate innestate fra loro ed essere in grado tornare a punti prestabiliti del programma ove richiesto. Ogni chiamata ad una funzione viene tradotta in una istruzione di ti-

WebDAV... un altro caso di overflow

Il recente bug della libreria di sistema NTDLL.DLL di Windows 2000 (Server e Professional) che ha causato tante polemiche su IIS e WebDAV, è un altro caso di buffer overflow. Per maggiori dettagli dare un'occhiata a ioProgrammo n.69 del mese di Maggio.

Buffer
OverFlow: l'arma segreta degli hacker

po *CALL*, che ha l'effetto di congelare l'esecuzione del codice fino a quel punto, passando poi il controllo alla funzione chiamata, dopo aver salvato lo stato corrente e l'indirizzo di ritorno (*return address*), che servirà per riportare l'esecuzione nel punto in cui si era interrotta (il registro *EIP* è quello che punta sempre all'istruzione corrente). Da questo meccanismo si evince una prima considerazione importante: poiché l'indirizzo di ritorno viene salvato nello stack, una eventuale corruzione di tale area dati impedirà ad un qualsiasi programma di ritornare in uno stato consistente, con conseguente crash dell'esecuzione.

LO STACK IN PRATICA

Per capire meglio il funzionamento dello stack serviamoci di un esempio banale, considerando un programma C++ nel cui *main()* è presente una generica funzione *f()*, a cui vengono passati due parametri. Per osservare l'uso dello stack, inseriamo nella funzione un buffer di 10 byte che non verrà effettivamente utilizzato nel sorgente.

```
//listato L1.CPP
int f(int a, int b) {
char buf[10];
int ris;
ris=a+b;
return ris; }
void main() {f(3, 2);}
```

Per vedere cosa avviene nello stack da vicino, basta compilare con Visual Studio (usando il comando *CL L1.CPP*) il file *L1.CPP* e successivamente aprire, usando lo stesso Visual Studio, l'eseguibile *L1.EXE* generato, avendo l'accortezza di specificare come tipo di file l'estensione *.EXE*. In questa modalità il Visual C++ di Microsoft si comporta come un vero debugger e consente di navigare nelle istruzioni *Assembly* corrispondenti al codice C++ scritto: per avviare l'esecuzione step-by-step bisogna premere *F11* e scegliere dalla finestra *View / Debug Windows* l'opzione *Disassembly*. Il codice macchina che si ottiene compilando questo programma apparirà grosso modo come quello che segue, commentato in questa sede solo per completezza:

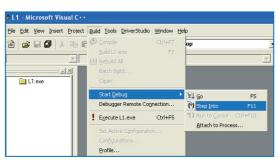



Fig. 3: Visual Studio integra un strumento di debugging avanzato, che può essere usato per aprire ed analizzare passo dopo passo i file eseguibili (EXE).

```
// chiamata alla funzione f() nel main
. . .
00401078push    2  //passaggio del parametro 2
0040107Apush    3  //passaggio del parametro 3
0040107C call  @ILT+0(f) (00401005)    //chiamata di f()
. . .
//funzione f()
00401020push  ebp   //salva il valore di EBP
00401021mov ebp,esp   //allinea ebp
00401023sub        esp,50h
00401026push        ebx  //salva lo stato dei registri
00401027push        esi
00401028push        edi
. . .
00401038mov     eax,dword ptr [ebp+8]
                              //carica il primo numero
0040103Badd      eax,dword ptr [ebp+0Ch]
                         //esegue la somma del secondo
0040103Emov       dword ptr [ebp-10h],eax
00401041mov      eax,dword ptr [ebp-10h]
                          //memorizza il risultato in EAX
00401044pop  edi   //ripristina i registri salvati
00401045pop        esi
00401046pop        ebx
00401047mov  esp, ebp         //ripristina lo stack
00401049pop         ebp
0040104Aret          //ritorna al chiamante
```

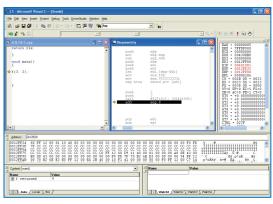



Fig. 4: Corrispondenza fra chiamata ad una generica funzione f() e il relativo codice Assembly (visto nel debugger).

Al momento della chiamata di *f()*, si vedono chiaramente le due istruzioni *PUSH* che memorizzano i valori "2" e "3" nello stack, mentre la *CALL* che invoca la funzione, ha come effetto quello di salvare nello stack l'offset corrente (*registro EIP*) e saltare all'indirizzo di *f()*. Una istantanea dello stack subito dopo la chiamata ad *f()* rivelerà questa situazione:

STACK

2
3
40107C (*return address*)
EBP
. . .
. . .

Shellcode

Uno shellcode è un particolare codice Assembly in grado di aprire una shell (per intenderci un *"cmd /c"*) sul sistema operativo. Per fare ciò, nei sistemi Windows, si deve individuare prima l'indirizzo di una particolare API di sistema, la *WinExec* o in alternativa la *CreateProcess*. Per localizzare tali indirizzi, si deve conoscere il kernel base address del sistema operativo Windows in uso, indirizzo che varia da sistema in sistema e che rende molti exploit, di fatto, non trasportabili su tutti i sistemi Microsoft:

WINDOWS PLATFORM	KERNEL BASE ADDRESS
Win95	0xBFF70000
Win98	0xBFF70000
WinME	0xBFF60000
WinNT (Service Pack 4 and 5)	0x77F00000
Win2000	0x77F00000

Un rimedio che permette di conoscere gli indirizzi del sistema è quello di localizzare la chiamata *GetProcAddress*, che consente di risalire a qualsiasi altra API del sistema.

IL PRIMO OVERFLOW

Una volta capito questo meccanismo, siamo pronti a passare all'analisi di un vero overflow dello stack. Consideriamo il seguente listato, *L2.CPP* :

```
//listato L2.CPP
#include <stdio.h>
void main(int argc, char **argv) {
char buf[20];
FILE* f=NULL;
int i=0;
printf("\nTest");
f=fopen("input.txt","rb");
while(!feof(f)) {buf[i]=fgetc(f); i++;}
//per sperimentare l'overflow basta creare un
//file "input.txt" con la seguente stringa di (20 + 8 bytes)
//AAAABBBBCCCCDDDDEEEEFFFFGGGG
```

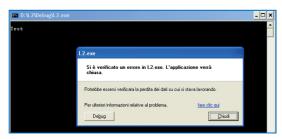



Fig. 5: Crash da overflow.

Si tratta di un esempio tipico di buffer overflow, dovuto all'uso incauto di un buffer nella lettura da file. Il codice non fa altro che leggere un carattere alla volta dal file *"input.txt"* fino alla fine per copiarlo dentro la variabile *buf*. Naturalmente, poiché non c'è alcun controllo sulla dimensione del file, basta superare il limite del buffer (20 byte) per provocare la condizione di overflow. Nell'esempio vengono aggiunti di proposito 8 byte nel file *"input.txt"*, proprio per causare un overflow dello stack; cliccando su *Debug* dopo il crash, si potranno notare alcune cose molto interessanti:

- l'esecuzione raggiunge una zona "indefinita", in cui non ci sono istruzioni valide (segnalate da "???");
- il valore di *EIP* è *0x47474747* (che corrisponde in hex alla stringa *"GGGG"*);
- il valore di *EBP* è *0x46464646* (che corrisponde in *hex* alla stringa *"FFFF"*).

In definitiva, ci accorgiamo che i valori presenti nell'input sono andati a sovrascrivere parte dello stack, sovrapponendosi così ai dati salvati in esso, fra cui l'indirizzo di ritorno (*return address*) e il base pointer (*EBP*).

CONTROLLARE IL FLUSSO DELL'ESECUZIONE

Modificare il registro *EIP* significa, in parole povere, riuscire a modificare l'esecuzione del programma, dirigendolo in una qualsiasi zona a nostro piacere. Una volta chiarito questo punto, possiamo iniziare a progettare un exploit, in grado di sfruttare l'overflow individuato per prendere il controllo del programma. Si procede studiando l'esecuzione del programma fino alla condizione di overflow, passo dopo passo, grazie alle funzioni di debugging offerte da Visual Studio. Ogni hacker che si rispetti è, infatti, costretto a disassemblare un eseguibile affetto dall'overflow prima di poter scrivere un exploit valido in grado di violarne la sicurezza. Osserviamo che la parte iniziale del codice Assembly non rappresenta il nostro programma ma si tratta di routine pre-impostate inserite dal compilatore di Visual C++. Il main vero e proprio inizia con la chiamata *CALL 401000* all'indirizzo

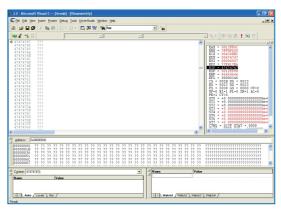



Fig. 6: Entrando in modo Debug dopo l'overflow, si può vedere come il registro EIP e il registro EBP hanno assunto i valori 0x46464646 e 0x47474747, che corrispondono alle stringhe di testo "FFFF" e "GGGG", passate in input.

```
EIP=401000          ESP=12FF84          EBP=12FFC0
00401000 push ebp     //salva EBP nello stack
00401001 mov          ebp,esp
00401003 sub  esp,1Ch          //riserva spazio per "buf"
00401006 mov dword ptr [ebp-18h],0 //variabile "FILE* f"
0040100D mov dword ptr [ebp-1Ch],0 //variabile "int i"
EIP=401014          ESP=12FF64          EBP=12FF80
```

Questa parte iniziale di codice viene generata dal compilatore e si "preoccupa" di riservare spazio nello stack utile per allocare buf (*0x1C byte*) e le altre variabili. Sono riportati i valori dei registri dello stack *ESP* e *EBP* prima e dopo l'esecuzione delle istruzioni. Successivamente il programma esegue la stampa a video con *printf()l* della stringa *"Test"*. In linguaggio Assembly i parametri di una funzione vengono passati mediante salvataggio nello stack: prima della chiamata a *printf()* troviamo infatti un'istruzione *PUSH* che memorizza nello stack l'offset della stringa di testo.

```
00401014 push      407030h       //offset string "\nTest"
00401019 call      00401114      //printf()
0040101E add       esp,4
...
```

Analizziamo adesso il ciclo di *while* che legge fino alla

Sicurezza

Buffer
OverFlow: l'arma segreta degli hacker

Login? ...no grazie!

Quanto mostrato in questo articolo può essere ovviamente visto nell'ottica della sicurezza su Internet; i servizi e i demoni dei server altro non sono che programmi scritti in C++. Nel corso della storia è capitato diverse volte che un telnet o un server Web/FTP sia risultato vulnerabile ad un buffer overflow, a causa di un username di lunghezza esagerata. Sfruttando questa forma di aggressione un hacker può entrare in un sistema ed eseguire comandi senza bisogno neanche di autenticarsi e di conoscere una password!

Sicurezza

Buffer
OverFlow: l'arma segreta degli hacker

Tipi di overflow

Oggi esiste una grande varietà di overflow nei programmi; sicuramente il più conosciuto e facile da manipolare per gli hacker è l'overflow dello stack, che permette di modificare in maniera efficace i valori dei registri EBP, ESP e EIP. Altri overflow, meno rinomati e più difficili da gestire, sono l'Heap Overflow, il Frame Pointer Exception e l'Adjacent Memory Overflow.

Bibliografia

Per chi fosse interessato all'intera storia del worm creato da Morris jr. nel 1988, consigliamo la lettura del seguente articolo:

- **UNIX OPERATINE SYSTEM SECURITY** *F.T. Grampp e R. Morris* **(AT&T Bell Lab. Technical Journal) Vol. 63, N. 8 Ottobre 1984**

Anche il sito http://world.std.com/~franl/worm.html **offre spunti interessanti sui dettagli della vicenda, illustrando in maniera dettagliata il funzionamento del worm.**

fine del file i byte memorizzandoli nella variabile buf:

```
00401033 mov      dword ptr [ebp-18h],eax
00401036 mov      eax,dword ptr [ebp-18h]
00401039 mov      ecx,dword ptr [eax+0Ch]
0040103C and      ecx,10h
0040103F test     ecx,ecx    //test di fine file (EOF)
00401041 jne      00401061
00401043 mov      edx,dword ptr [ebp-18h]
00401046 push     edx
00401047 call     004010C7 //lettura da file fgetc()
0040104C add      esp,4
0040104F mov      ecx,dword ptr [ebp-1Ch]
00401052 mov      byte ptr [ebp+ecx-14h],al
                           //memorizza il byte in buf
00401056 mov      edx,dword ptr [ebp-1Ch]
00401059 add      edx,1               //i++
0040105C mov      dword ptr [ebp-1Ch],edx
0040105F jmp      00401036            //ciclo
```

L'istruzione alla riga *401052* è quella che scrive nel buffer il dato letto da file mediante *fgetc()*. L'indirizzo del buffer è dato da *[EBP+ECX-14h]*; il valore viene incrementato ad ogni iterazione grazie al registro *ECX*. È tuttavia la parte finale del programma, quella che segue immediatamente all'istruzione *fclose()*, la più interessante: viene ripristinato il valore del registro *EBP* e per chiudere la procedura si esegue un'istruzione di ritorno *RET*. Tale istruzione ha l'effetto di estrarre una word (32-bit) dallo stack e di memorizzarla in EIP, modificando l'indirizzo dell'istruzione corrente e spostando così il flusso di esecuzione del programma. È in questo momento che emergono i problemi dell'overflow e il programma raggiunge una condizione indefinita. Il ciclo di *while* infatti, non si accorge di scrivere nel buffer più dati di quelli previsti (28 contro 20) e di conseguenza inizia a scrivere sopra lo stack alterando i valori in esso memorizzati.

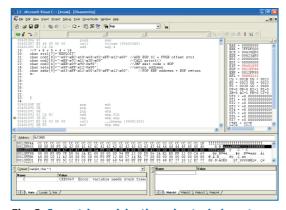



Fig. 8: **Incontri ravvicinati con lo stack durante l'overflow: si vede chiaramente il punto in cui l'input passato al programma trabocca e sovrascrive altre zone di memoria.**

```
00401061   mov    eax,dword ptr [ebp-18h]
00401064   push   eax
00401065   call   00401071   //fclose()
0040106A   add    esp,4
0040106D   mov    esp,ebp
0040106F   pop    ebp              //ripristina EBP
00401070   ret                     //istruzione di rit.
```

Tutto ciò si traduce con un errore che scatta nel momento in cui si eseguono le istruzioni *POP* e *RET*: i valori estratti dallo stack non sono più quelli corretti, ma sono diventati parte dei dati letti da input; infatti, nel momento in cui si verifica il crash, ci si accorge che i registri *EBP* e *EIP* contengono i valori *"0x46464646"* e *"0x47474747"*, che corrispondono proprio ai caratteri in eccedenza presenti nel file *input.txt* (rispettivamente le stringhe *"FFFF"* e *"GGGG"*).

EXPLOIT!

Come si realizza un exploit capace di sfruttare l'overflow appena visto? Abbiamo mostrato come sia possibile controllare l'andamento del programma fornendo input in eccedenza, ma possiamo fare di più e spingerci oltre: se nell'input invece di fornire caratteri e stringhe di testo, memorizziamo istruzioni Assembly a nostro piacimento, possiamo iniettare il nostro codice direttamente nello stack del programma e quindi modificare l'andamento del flusso d'esecuzione, dirottando il registro EIP nel punto in cui si trova il codice iniettato. L'unica difficoltà di questa fase è il calcolo degli indirizzi e degli offset di allineamento, che dovranno combaciare perfettamente per far sì che il programma ad un certo punto esegua le nostre istruzioni; analizzando la stringa di overflow ci si rende conto che il valore GGGG pilota il registro EIP.

AAAA	*BBBB*	*CCCC*	*DDDD*	*EEEE*	*FFFF*	*GGGG*
4	*8*	*12*	*16*	*20*	*registro EBP*	*registro EIP*

La struttura dell'exploit prevede, quindi, un input di questa forma: in testa possiamo sfruttare i primi 20 byte del buffer per memorizzare il codice che vogliamo far eseguire, aiutandoci con eventuali istruzioni *NOP* (istruzione Assembly ininfluente, di *no-operation*) per allineare il codice esattamente alla dimensione di 20 byte. Seguono due word da 32-bit che corrispondono ai valori scritti dall'overflow nei registri *EBP* e *EIP;* infine, poiché per questo exploit si è pensato di visualizzare una stringa di testo, memorizzeremo nella parte finale dell'input una stringa di testo, terminata da 0.

INJECTED CODE	NOP (0x90)	EBP	EIP	STRING\0
0…	…20	1 word (4 byte)	1 word (4 byte)	"EXPLOIT WORKS"

Analizzando col debugger l'esecuzione del programma, si nota che il valore di *EBP* prima dell'overflow è *0x0012FF80*, mentre lo stack usato per memorizzare i dati inizia all'offset *0x0012FF6C*, che sarà proprio il punto in cui partirà il codice iniettato. Conosciamo

quindi i valori delle word di *EBP* e di *EIP*, non resta che scrivere il codice da iniettare. Lo scopo di un hacker è quello di progettare uno *shellcode*, cioè un codice capace di aprire una shell remota in grado di accettare comandi dall'esterno; la scrittura degli shellcode è diventata ormai un'arte, a causa di numerose problematiche ad essa legate. Primo fra tutti c'è il problema della differenza fra sistemi operativi (uno shellcode per Linux non funziona su Windows), anche quando si tratta della stessa famiglia (gli indirizzi base del kernel di Windows variano a seconda della versione). Altra difficoltà è rappresentata dalla codifica Unicode delle stringhe di testo (che altera il codice iniettato dall'hacker) e dai sistemi anti-intrusione IDS (che identificano gli shellcode come con i virus e costringono gli hacker a scrivere shellcode criptati). Nel nostro esempio non scriveremo uno shellcode, poiché l'utilizzo di tale tecnica richiederebbe un articolo a sé stante, per cui ci limiteremo a far eseguire una istruzione *printf()* che visualizzi a stringa iniettata *"EXPLOIT WORKS"*. Cos'altro serve per scrivere il codice? Naturalmente abbiamo bisogno dell'offset della stringa da visualizzare e dell'indirizzo di *printf()*, che – come abbiamo visto prima – è localizzata a *0x0040 1114*. Il codice Assembly da iniettare sarà così fatto, per un totale di *3 + 5 + 5 = 13* byte.

OPCODES	ISTRUZIONE
83 EC 20	SUB ESP, 0x20
68 x1 x2 x3 x4	PUSH offset(stringa)
E8 y1 y2 y3 y4	CALL printf

Per raggiungere i 20 bytes richiesti dall'overflow, si aggiungono 7 istruzioni NOP alla fine del codice. Gli opcodes sono i codici esadecimali che corrispondono alle istruzioni Assembly; ad esempio "83 EC" identifica l'istruzione "SUB ESP", che seguita dal valore 0x20, sottrae 20 bytes dal registro ESP. Non rimane che calcolare i valori "x1 x2 x3 x4" e "y1 y2 y3 y4" della PUSH e della CALL. L'offset della stringa di testo si calcola partendo dall'indirizzo iniziale del nostro buffer (dove inizia il codice, 0x0012FF6C) a cui si aggiungono i 28 bytes del buffer, ottenendo 0x0012FF88. Per calcolare il valore di *"y1 y2 y3 y4"* occorre invece partire dall'indirizzo della funzione *printf() 0x00401114* a cui bisogna sottrarre l'offset in cui si trova l'istruzione *CALL*, che è dato da *0x0012FF6C+3+5+5 = 0x 0012FF79*. Quindi otteniamo per *"y1 y2 y3 y4"* il valore di *0x002D119B*. Riscrivendo il codice avremo ora:

OPCODES	ISTRUZIONE
83 EC 20	SUB ESP, 0x20
68 88 FF 12 00	PUSH 0x0012FF88
E8 9B 11 2D 00	CALL 0x002D119B
90	NOP
90	NOP
90	NOP
90	NOP
90	NOP
90	NOP
90	NOP

Piccolo accorgimento: gli indirizzi e gli offset delle istruzioni Assembly vanno scritti al contrario, cioè leggendoli da destra verso sinistra. Per creare un file di input fatto in questa maniera, basta ricorrere a un semplice programma C++, che unisca i vari pezzi dell'exploit, scrivendoli su un unico file *"input.txt"* sul CD Rom trovate il listato *Gen.CPP*.
Usando come input il file generato da *GEN.CPP* per il programma del listato *L2.CPP*, si realizza l'exploit che prende il controllo del programma in fase di esecuzione e visualizza la stringa *"EXPLOIT WORKS"* sullo schermo. Ovviamente al termine della *printf()* il programma si trova in uno stato non definito e va comunque in crash, anche se tecnicamente era possibile portare a termine l'esecuzione senza visualizzare alcun errore, dirottando il codice verso la chiamata ad *exit()*.

CONCLUSIONI

Prevenire è meglio che correre ai ripari. Ma come ci si può difendere dagli overflow? Esistono diverse soluzioni proposte dagli esperti, alcune delle quali richiederebbero la modifica dei kernel e dei principali compilatori.
Alcune di queste sono:

- **Non Executable Stack -** Modificando il kernel si può rendere il segmento che contiene i dati dello stack non-eseguibile, in modo che se accidentalmente o volutamente un programma dovesse finire in questa zona di memoria, verrebbe generato un *protection fault*, terminando il processo.
- **Random Stack Address -** Per poter pilotare il flusso di esecuzione, un exploit deve disporre di indirizzi affidabili e validi da assegnare ai registri *EIP* e *ESP*. Utilizzando uno stack con modalità di indirizzamento casuale, si potrebbe complicare ulteriormente la vita degli hacker nella realizzazione di exploit di portata universale.
- **Bounds Checking -** Il problema degli overflow nasce dalla mancanza di controlli da parte del programmatore e del compilatore. Piccoli accorgimenti possono fare la differenza, come in questo caso:

```
strcpy(buf, ptr);              // insicuro!!!
strncpy(buf, sizeof(buf), ptr);          // sicuro
```

Le funzioni standard del C++ vulnerabili agli overflow sono comunque tante, tra cui le principali individuate sono: *strcopy(), strcat(), sprintf(), gets(), scanf()*. Il controllo dei limiti può essere effettuato dal programmatore, ma potrebbe essere gestito anche a livello di librerie, usando funzioni *safe*.

Ing. Elia Florio

Sicurezza

Buffer
OverFlow: l'arma segreta degli hacker

Sul Web

Protecting against some buffer-overrun Attacks – R. Kettlewell, 1998
http://www.greenend.org.uk/rjk/random-stack.html

Buffer Overflows : Defending against arbitrary code insertion and execution – S. Fewer
http://www.harmonysecurity.com/paper_bufferoverflow.html

Smashing the Stack for fun and profit – Phrack Magazine nr.49 – Aleph One
http://www.phrack.org/show.php?p=49&a=14

Analisys of Buffer Overflow attacks
http://www.windowsecurity.com/articles/Analysis_of_Buffer_Overflow_Attacks.html

Non-stack Overflows in Windows – David Litchfield
http://www.nextgenss.com/papers/non-stack-bo-windows.pdf

☑ In pratica: protocolli di comunicazione.

Un mail client in Visual Basic

Posta elettronica con VB

Lo scopo di questa serie di articoli è arrivare a realizzare un completo client di posta elettronica. In questo primo appuntamento analizzeremo i protocolli di comunicazione.

Tra i servizi offerti da Internet, quello che probabilmente risulta essere il più utilizzato, è quello della posta elettronica. Internet è ormai diventata una realtà da moltissimi anni e questo servizio per il recapito e la ricezione di mail elettroniche può essere considerato, senza ombra di dubbio, uno dei principali motivi si successo.Sappiamo tutti che un indirizzo di posta elettronica generico è una stringa molto simile alla seguente: *username@host.dominio* in cui *username* è la sottostringa che identifica "genericamente" l'interlocutore al quale inviare il messaggio di posta elettronica, mentre *host. dominio* rappresenta un nome DNS o un indirizzo IP presso il quale l'utente "possiede" una casella di posta elettronica. Ovviamente questa sintetica descrizione racchiude in sé molto più di una semplice stringa poiché, come vedremo, le componenti (software e non) che entrano in gioco quando inviamo un messaggio di posta elettronica sono davvero tante. La gestione della posta elettronica è implementata in Internet attraverso la cooperazione di due categorie di sottosistemi:

- *Mail User Agent (MUA);*
- *Mail Transport Agent (MTA).*

Il *MUA* rappresenta il generico programma di gestione della posta (*Outlook, Eudora, Pegasus Mail*, ecc.) installato sul proprio client. Esso deve possedere alcune caratteristiche fondamentali:

- Interfaccia utente per la gestione dei messaggi di posta elettronica, ossia deve consentire la composizione, la ricezione e la lettura dei messaggi.
- Deve essere in grado di "colloquiare" con i mail server che provvedono all'invio dei propri messaggi all'MTA.
- Riconoscere la sintassi di composizione dei messaggi (secondo le specifiche RFC822 e MIME)

L'MTA possiamo vederlo invece come il canale di comunicazione tra due MUA. Il suo scopo principale è quello di consentire la ricezione di tutti i messaggi ed il loro recapito. L'MTA, secondo quanto già intuito, può essere:

- un server SMTP che si occupa della spedizione e della ricezione delle mail da e verso altri server SMTP;
- un server POP3 che si occupa di spedire i messaggi ad ogni client;
- un server IMAP4 che consente la gestione dei messaggi direttamente sul server.

Naturalmente, ognuno di questi servizi è attivo su una porta TCP/IP diversa, ognuno di essi potrebbero risiedere su di uno stesso server o su server diversi, cosa che accade, solitamente, in realtà aziendali molto articolate. In questo articolo vedremo come implementare un semplice client di posta elettronica in grado di leggere ed inviare mail. Per poter comprendere il funzionamento di questo programma (ed in generale, di ogni programma che implementa il colloquio con un server mediante un qualunque protocollo), è opportuno avere dimestichezza con i protocolli di comunicazione che si dovranno utilizzare ed essere in grado d'implementare un generico dialogo mediante Telnet. Questo consente di provare "in loco" la corretta implementazione del colloquio client-server nel linguaggio (protocollo) specificato ed individuare con mag-

RFC1425

Il documento RFC1425 descrive una nuova versione di SMTP denominata ESMTP (Extended SMTP). Il client che desidera farne uso invia, all'inizio della comunicazione con il server, la stringa EHLO anziché HELO.

Posta elettronica **con VB**

Un mail client *in Visual Basic*

giore facilità i problemi ai quali si può incorrere in fase di programmazione.

IL PROTOCOLLO SMTP

Il protocollo *SMTP* costituisce il "linguaggio" standard per l'invio delle proprie mail. Un generico esempio di comunicazione in SMTP è mostrato in Fig. 1.

```
Telnet SMTPSRV 25

220 SMTPSRV.MyDomain.it ESMTP Service (Lotus Domino Release 5.0.11) rea
HELO 10.0.1.100
250 SMTPSRV.MyDomain.it Hello 10.0.0.200 ([10.0.0.200]), pleased to meet you
MAIL FROM: flippo@MyDomain.it
250 flippo@MyDomain.it... Sender OK
RCPT TO: flippo@libero.it
250 flippo@libero.it... Recipient OK
RCPT TO: flippo@email.it
250 flippo@email.it... Recipient OK
DATA
354 Enter message, end with "." on a line by itself
FROM: Francesco Lippo (Ed Master)
TO: flippo@libero.it
CC: flippo@email.it
SUBJECT: TEST ED. MASTER

Prova invio mail per IoProgrammo
.
250 Message accepted for delivery

quit

Connessione all'host perduta.
```

Fig. 1: Un esempio di comunicazione in SMTP con un server di posta.

I comandi principali di questo protocollo sono:

- **HELO**: questo comando dà inizio al colloquio con il server SMTP e serve ad identificare il client, mediante il parametro *Hostname*. Un generico comando è: HELO *flippo*.
- **MAIL FROM:** con questo comando è inizializzata la vera e propria transazione. In questo caso si deve indicare il mittente (*reverse-path*) a cui è riferita la creazione della mail. Un generico formato del comando è MAIL FROM: *Francesco.Lippo@edmaster .it*. Il server risponde con un reply number 250 del tipo *Sender OK*. Questo comando pulisce i buffer e inserisce nello stesso buffer dedicato al mittente, la mail del client.
- **RCPT TO**: serve ad identificare i destinatari del messaggio. Un generico esempio è RCPT TO: *Francesco.Lippo@edmaster.it*.
- **RSET**: consente di "resettare" le impostazioni correnti dell'email che si stava configurando per l'inoltro, permettendo di ricominciare con un nuovo messaggio di posta elettronica.
- **DATA**: lanciato senza parametri, offre la possibilità d'inserire il testo della nostra mail che dev'essere necessariamente completato da questa sequenza: *<CRLF>. <CRLF>*. Se tutto è andato a buon fine, il server risponde con un messaggio di esito positivo in cui si indica che il messaggio è stato accettato per l'invio.
- **VRFY**: consente di verificare l'email di un qualunque destinatario di posta elettronica. Un esempio è *VRFY Francesco.Lippo@edmaster.it*.
- **HELP**: mostra l'elenco dei comandi accettati dal server.
- **QUIT**: chiude il colloquio tra client e server SMTP.

Il documento RFC1425 descrive una nuova versione di SMTP denominata *ESMTP (Extended SMTP)*. Il client che desidera farne uso invia, all'inizio della comunicazione con il server, la stringa *EHLO* anziché *HELO*. Se il server con cui si sta comunicando supporta ESMTP, la risposta inviata al client sarà un certo numero di righe con prefisso/codice 250. Questa risposta è di solito multilinea, poichè ognuna contiene una parola chiave e, opzionalmente, un argomento che specifica le estensioni SMTP supportate dal server. In caso contrario, ossia qualora il server non riconoscesse il comando *EHLO* (vale a dire, quindi, le estensioni ad SMTP), invierà al client un codice di errore. Non ci soffermeremo a lungo su questo protocollo, ma come già detto è consigliabile effettuare delle prove con un server presso il quale si possegga una casella di posta (anche locale), mediante Telnet sulla porta 25, per "simulare" l'invio di un messaggio di posta tramite esso.

IL PROTOCOLLO POP3

Contrariamente a quanto si potrebbe pensare, il protocollo per la ricezione dei messaggi di posta elettronica è diverso da quello impiegato per l'invio. Il più utilizzato è senza dubbio il *Post Office Protocol 3* (POP3), ma può essere sfruttato anche l'IMAP4. In realtà esistono più versioni di questo protocollo (il numero 3 accanto all'acronimo POP indica, per l'appunto, la terza versione di questo protocollo), ma la versione POP3 è senza dubbio quella più sfruttata. L'implementazione del protocollo POP3 è relativamente più complessa di quella impiegata per SMTP. Una generica conversazione in POP3 prevede essenzialmente tre fasi distinte:

- **Autenticazione**: in questa fase, il client invia al server le proprie credenziali (*username* e *password)* per richiedere l'accesso alla propria casella postale. Qualora la risposta del server sia positiva, quest'ultimo ne concede

Protocolli

Contrariamente a quanto si potrebbe pensare, il protocollo per la ricezione dei messaggi di posta elettronica è diverso da quello impiegato per l'invio. Il più utilizzato è senza dubbio il *Post Office Protocol 3* (POP3), ma può essere sfruttato anche l'IMAP4.

Posta elettronica
con VB

Un mail client
in Visual Basic

l'accesso, bloccandone l'utilizzo da parte di altri programmi.

- **Transizione**: durante questa fase, il client scarica effettivamente i messaggi dalla propria casella postale. L'utente termina questa conversazione con il comando *QUIT*.
- **Aggiornamento**: durante questa fase il server procede ad effettuare un aggiornamento delle informazioni, eliminando gli eventuali messaggi contrassegnati per l'eliminazione nella *Transaction Phase*, sbloccando la casella postale ed infine chiudendo la connessione definitivamente.

Analogamente a quanto accadeva con SMTP, ogni utente che desidera scaricare i propri messaggi di posta elettronica, deve stabilire una connessione TCP sulla porta 110 del server POP3. Subito dopo aver stabilito la connessione, il server si "presenta" ed è pronto ad accettare i comandi ad esso inviati. Come abbiamo avuto modo di accennare, la prima fase prevede l'autenticazione dell'utente. Il protocollo POP3 prevede diversi modi con cui il client può identificarsi, ma quello più utilizzato consiste nell'utilizzo dei comandi *USER* e *PASS*. Attraverso il primo, il client invia al server il nome utente registrato corrispondente ad una casella postale ben definita. Se il server accetta il nome allora è possibile spedire anche la password mediante il comando *PASS*, seguito dall'opportuna parola d'ordine associata al nome in questione. A questo punto, avvenuta l'autenticazione, il client ha finalmente accesso alla propria casella postale e può, quindi, effettuare operazioni su di essa. Ha inizio la seconda fase del collegamento (*Transaction Phase*). Durante lo scambio delle informazioni tra client e server, è possibile osservare due soli possibili indicatori di stato che, posti all'inizio di ogni riga, identificano immediatamente la risposta del server, oltre ad un'azione richiesta dal client: *+OK* e *–ERR*. Una volta entrati nella fase di transazione, il server assegna a ogni messaggio un identificativo numerico (partendo dal numero uno) mentre il client può iniziare, a sua volta, la spedizione delle proprie richieste utilizzando una serie di comandi a sua disposizione. Ecco i principali:

I Reply Number di un server SMTP

Di seguito è mostrato un elenco dei cosiddetti *Reply Number* ossia i messaggi possibili che il server *SMTP* restituisce al client quando quest'ultimo gli invia un qualunque comando. Accanto ad ognuno è mostrata una breve descrizione sul loro significato:

211	System status, o system help reply.
214	Messaggio di Help.
220	<domain> Servizio pronto.
221	<domain> Servizio ha chiuso il canale.
250	Azione completata, OK.
251	Utente non locale; si spedirà a <forward-path>.
252	Non è possibile verificare l'email, ma è comunque accettata.
354	Inizia l'input dei dati; termina con <CRLF>.<CRLF>.
421	Servizio non avviabile.
450	Richiesta di azione non avviabile.
451	Richiesta di azione abortita: errore locale durante il processo.
452	Richiesta di azione abortita: spazio di sistema insufficiente.
500	Errore di sintassi, comando non riconosciuto.
501	Errore di sintassi nei parametri o negli argomenti.
502	Comando non implementato.
503	Cattiva sequenza del comando.
504	Parametri del comando non implementati.
550	Richiesta di azione non presa.
551	Utente non locale; tentare con <forward-path>.
552	Richiesta di azione abortita: si eccede l'allocazione di spazio.
553	Richiesta di azione non presa: nome della mailbox non permesso.
554	Transazione fallita.

Dowload messaggi di posta

Analogamente a quanto accadeva con SMTP, ogni utente che desidera scaricare i propri messaggi di posta elettronica, deve stabilire una connessione TCP sulla porta 110 del server POP3.

```
Telnet POP3SRV 110
+OK POP3 PROXY server ready (6.5.001) <371E4DCC077DEE7F726AAC5A1D8680473BBCBB
>
USER flippo
+OK Password required
PASS pippo
+OK 27 messages
RETR 27
+OK 1626 bytes
Return-Path: <flippo@MyDomain.it>
Received: from smtp7.MyDomain.it (193.70.192.90) by ims2c.MyDomain.it (7.0.012)
      id 3E933ED3000E47E8 for flippo@MyDomain.it; Fri, 11 Apr 2003 09:33:57 +0200
Received: from it-hubm.MyDomain.it (212.141.56.78) by smtp7.MyDomain.it (6.7.015)
      id 3E8DC05700CE9AD7 for flippo@MyDomain.it; Fri, 11 Apr 2003 09:33:57 +0200
Received: from POP3SRV.MyDomain.it ([138.70.193.142])
        by it-hubm.MyDomain.it (Lotus Domino Release 5.0.11)
        with ESMTP id 2003041109354487:73 ;
        Fri, 11 Apr 2003 09:35:44 +0200
Received: from 138.70.192.180 ([138.70.192.180])
        by POP3SRV.MyDomain.it (Lotus Domino Release 5.0.11)
        with SMTP id 2003041109325928:23 ;
        Fri, 11 Apr 2003 09:32:59 +0200
FROM: Francesco Lippo Ed Master
CC: flippo@email.it
X-MIMETrack: Itemize by SMTP Server on POP3SRV/IT/MyDomain(Release 5.0.11 |July 24, 20
 11/04/2003 09.34.28,
      Serialize by Router on POP3SRV/IT/MyDomain(Release 5.0.11 |July 24, 2002) at
 11/04/2003 09.34.29,
      Serialize complete at 11/04/2003 09.34.29,
      Itemize by SMTP Server on IT-HUBM/IT/MyDomain(Release 5.0.11 |July 24, 2002) at
 04/11/2003 09:35:44 AM,
      Serialize by Router on IT-HUBM/IT/MyDomain(Release 5.0.11 |July 24, 2002) at
 04/11/2003 09:35:46 AM,
      Serialize complete at 04/11/2003 09:35:46 AM
To: flippo@Mylibero.it, flippo@email.it
Date: Fri, 11 Apr 2003 09:34:28 +0200
Message-ID: <OF51E91996.06D52D06-ONC1256D05.00299BE5@MyDomain.it>

SUBJECT: TEST ED. MASTER

Prova invio mail per IoProgrammo
.
LIST
+OK
1 1871
2 20836
3 24317
...
26 1542
27 1626
.
DELE 27
+OK message marked for deletion
LIST
+OK
1 1871
```

Fig. 2: Un esempio di comunicazione in POP3 con un server di posta.

- **RETR**: questo comando, seguito dal numero di messaggio, consente di scaricare una generica mail e leggerla.

- **STAT**: restituisce il numero di messaggi presenti nella propria casella postale e la dimensione totale della stessa.

- **LIST**: restituisce l'elenco delle mail contenute all'interno della casella postale, numerate in ordine progressivo e la dimensione di ciascuna.

- **NOOP**: restituisce semplicemente *+OK*. Questo semplice comando, poco utilizzato, può servirci a controllare lo "stato" del server per assicurarci che esso sia ancora raggiungibile ed in ascolto.

- **DELE**: contrassegna per la cancellazione un determinato messaggio.

- **RSET**: reimposta il contrassegno che identifica, per ogni mail, la richiesta di eliminazione dal server.

In Fig. 2 è mostrato un generico colloquio in POP3 con il proprio server di posta. Per ragioni di spazio, la lista dei messaggi presenti all'interno della casella di posta usata per i test (al momento delle prove risultava un totale di 27 messaggi), è stata "sintetizzata" mostrando semplicemente le prime e le ultime mail. Si osservi un particolare importante, ossia il fatto che la password relativa alla propria casella postale, digitata dopo *PASS*, è inviata "in chiaro". Quest'annotazione può essere importante qualora si decida di realizzare un colloquio in Telnet con il proprio server di posta e non si desideri far conoscere ad altri quest'informazione. Quindi, in definitiva, è bene essere certi che nessuno ci sia accanto quando la digitiamo per inviarla al nostro server POP3. Prima di passare oltre, è bene tener presente alcuni comandi ai quali bisognerebbe prestare molta attenzione: *RETR, DELE* e *RSET.* Il primo, permette di ricevere un messaggio e richiede come unico parametro il numero del messaggio stesso. L'output a video è molto simile a quanto mostrato in Fig. 2 e, a meno di non essere nel caso di operazioni particolari, l'unica parte importante è quella relativa al mittente ed al body dell'email. Il secondo comando, invece, permette di marcare un messaggio della casella postale affinché possa essere cancellato durante la fase di aggiornamento. Sottolineo "marcare" poiché, dopo quest'operazione, tutti i messaggi contrassegnati attraverso *DELE* non risultano ancora essere stati eliminati dal server, pur risultando nascosti ed apparentemente eliminati. Sarebbe opportuno lanciare questo comando dopo ogni lettura, in maniera tale da liberare spazio all'interno della propria casella postale. Comunque sia, qualora ci si accorga di aver lanciato erroneamente il comando *DELE*, basterà servirsi di *RSET* per "resettare" la lista delle mail da cancellare.

IL FORMATO MIME

In questo paragrafo parleremo brevemente di MIME al fine di avere un quadro più chiaro delle problematiche che coinvolgono un servizio di posta elettronica. Inizialmente il protocollo per la rappresentazione dei documenti di posta elettronica era definito nel documento RFC 822. Esso specificava il formato per i messaggi di posta, limitandosi a messaggi esclusivamente di tipo testo ASCII, senza alcun riferimento ad altri formati.

Una delle principali limitazioni del protocollo descritto in *RFC 822* risiede nel fatto che il contenuto dei messaggi è limitato ai soli caratteri codificati in 7 bit. Quest'imposizione obbliga, prima dell'invio, a convertire ogni messaggio che non contenga esclusivamente semplice testo ASCII. Per ovviare a questo inconveniente, nel 1992 fu finalmente introdotto un nuovo documento nel quale veniva descritto lo standard *MIME (RFC 1341)*. In esso vengono presentati i meccanismi per superare le limitazioni contenute in RFC 822 ossia:

- definizione del formato di messaggi testuali (ASCII e non)

- definizione del formato di messaggi multimediali (contenenti video, suono, immagini, ecc.).

Il MIME e, nello specifico, *S/MIME*, è stato decisivo per l'implementazione dei servizi di sicurezza che offrono all'utente la possibilità di inviare messaggi corredati di firma digitale, di crittografia o di autenticazione.

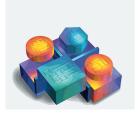
Qualunque linguaggio di programmazione si desideri utilizzare, per poter implementare correttamente un client di posta, occorre conoscere necessariamente le specifiche appena menzionate. L'*RFC 822* è costituito "semplicemente" da una stringa di testo formata da due parti distinte: un *header* ed un *body*, separati da una linea vuota. L'*header*, che racchiude in sé le informazioni per il trasporto della mail, può essere così schematizzato:

- **TO**: indica la lista dei destinatari.

- **FROM**: mittente dell'email.

Gli standard per la posta elettronica

Gli standard ed i protocolli fondamentali per il funzionamento della posta elettronica in ambiente Internet sono:

- **il formato dei messaggi di posta, descritto nell'RFC 822;**
- **il formato MIME per la definizione del formato dei dati, descritto negli RFC 1341, 1342, 2045, 2046, 2047 e 2048 e che estende il formato descritto nell'RFC822;**
- **il protocollo SMTP, descritto nell'RFC 821;**
- **il protocollo POP3, descritto nell'RFC 1939.**

Posta elettronica
con VB

Un mail client
in Visual Basic

- **CC**: lista dei destinatari per conoscenza.
- **BCC**: lista nascosta di destinatari per conoscenza.
- **DATE**: data dei spedizione dell'email.
- **REPLY-TO**: indirizzo alternativo del mittente a cui inviare le risposte.
- **SUBJECT**: oggetto del messaggio.

Con lo standard MIME è possibile inserire, in un qualsiasi messaggio e-mail, oltre al testo, anche file contenenti immagini, segnali audio e video. In particolare, è bene ricordare che il software che dovrà gestire la posta non deve preoccuparsi del contenuto del messaggio, poichè è l'utilizzatore finale a dover applicare l'opportuna decodifica in base alle "specifiche di tipo" inserite nel messaggio stesso. Un documento MIME contiene una testata in cui si trovano i seguenti campi:

- **MIME version**: identifica la versione dello standard MIME usato nella mail.
- **Content-Transfer-Encoding**: specifica un modo di codifica dei dati accessorio a quello principale. I possibili valori che può assumere questo attributo sono *7bit*, *8bit*, *binary*, *quoted-printable* e *base64*.
- **Content-Type**: consente di specificare il tipo ed il sottotipo di dati contenuti all'interno del messaggio *(MIME type)*. Attraverso esso, il client che riceve la mail è in grado di rilevare in che maniera sono stati codificati i dati ricevuti. Questo attributo, insieme a quello precedente, identificano la parte fondamentale di questo standard e, nel contempo, l'aspetto più complesso da gestire.
- **Content-ID**: identifica il messaggio in modo univoco. Questo attributo è opzionale.
- **Content-Description**: rappresenta una descrizione testuale del contenuto del messaggio. Anche questo attributo è opzionale.

Per non complicare troppo questa introduzione ai servizi di posta elettronica, è stato volutamente omesso di descrivere più in dettaglio questo standard, lasciando ai più volenterosi il compito di approfondire l'argomento.

Uuencode

Uuencode **presuppone che il file d'input sia di tipo binario. Stabilito questo, estrae dal file binario 3 byte per volta e li converte in 4 caratteri ASCII.**

LA CODIFICA UUENCODE

Sappiamo benissimo che un mail client qualunque consente di aggiungere degli allegati al nostro messaggio, permettendoci anche di salvarli una volta ricevuta la mail. Gli allegati possono essere costituiti da file di qualsiasi genere. Ciò, ovviamente, significa anche che non è possibile inserirlo "immediatamente" all'interno del nostro messaggio poiché, come abbiamo visto, una generica mail è costituita "necessariamente" da testo puro. Un modo per risolvere il problema è quello applicare al nostro file un algoritmo tale da convertirlo in una forma che possa essere inclusa all'interno della struttura della mail, ma che non "interferisca" con il testo stesso. L'algoritmo che consente di ottenere tutto questo esiste ed è definito con il nome *uuencode*. *Uuencode* presuppone che il file d'input sia di tipo binario. Stabilito questo, estrae dal file binario 3 byte per volta e li converte in 4 caratteri ASCII. L'algoritmo utilizzato, visto più da vicino, consiste nel:

- Considerare i 3 byte come un'unica parola da 24 bit.
- Prendere 6 bit alla volta creando 4 nuovi byte.
- Aggiungere 32 ad ogni nuovo byte per ottenere un carattere di testo in formato ASCII.

Il metodo è molto semplice e sicuramente efficiente poiché consente di ottenere, al termine di questa elaborazione, un file contenente soltanto caratteri di testo, anche se un pò più grande di quello originario.
Di seguito è mostrato un piccolo esempio che mostra un messaggio decodificato secondo questa tecnica:

```
begin 664 Allegato.txt
%3TLN+BX@
'
end
```

Il valore 664 rappresenta un codice che consente d'indicare espressamente l'utilizzo della codifica Base64.

CONCLUSIONI

Ora che abbiamo gettato le basi per la comprensione di un sistema di posta elettronica, possiamo concludere questa puntata. La prossima volta vedremo praticamente come applicare i concetti appena visti in un progetto in Visual Basic, sfruttando nel contempo questa occasione per dare un'occhiata ad "altri" aspetti che potrebbero tornarci utili nella realizzazione di programmi simili.

Francesco Lippo e Alberto Lippo

Nmap... Reloaded!

Mossi da quanto visto sul grande schermo ultimamente (mi riferisco a Matrix Reloaded), ci dedicheremo alla presentazione di un tool molto famoso al mondo degli hacker, conosciuto col nome di Nmap.

Ogni hacker che si rispetti precede sempre la fase di intrusione vera e propria in una rete informatica con una fase iniziale di "esplorazione", nella quale viene studiata a fondo la subnet oggetto dell'attacco, cercando di raccogliere quante più informazioni possibili su di essa e sugli host presenti. Vista la natura delle vulnerabilità, che variano da applicativo ad applicativo (IIS, Apache, NetBIOS, BIND, ecc.) e considerato anche che gli shellcode, i codici per l'esecuzione remota di shell, inseriti negli exploit sono specifici per ogni sistema operativo (un exploit con shellcode per Linux Red Hat non funziona certo su SunOS!), appare evidente che la raccolta di quante più informazioni possibili sul bersaglio favorisce la riuscita di un attacco.

DIMMI CHE SISTEMA SEI...

Supponendo ad esempio di conoscere l'host target tramite il suo indirizzo IP (ipotizziamo ad esempio 192.168.1.30), la prima cosa da fare per raccogliere le informazioni che ci servono è quella di eseguire un ping-scan per trovare tutti gli host attivi sulla subnet 192.168.1.*, tenendo presente però che alcuni host potrebbero non rispondere ai messaggi ICMP magari perché protetti da un firewall. Questa prima fase esplorativa fornisce un elenco di possibili bersagli, utile perché spesso se un hacker non riesce a entrare dall'esterno su un certo host X, può cercare di bucare un altro host Y della stessa rete di X, che magari è autorizzato ad accedere ad X.
Una volta ottenuti gli host attivi, bisognerà cercare di capire quale sistema operativo essi montano e quali servizi e porte TCP/IP sono attive, perché proprio da una di queste porte l'hacker potrebbe riuscire ad entrare. Prima dell'avvento delle tecniche di OS Fingerprinting, basate sull'analisi dello stack e sulla produzione di pacchetti TCP/IP anomali, i metodi più utilizzati erano l'analisi dei "banner" e delle porte aperte, che in modo abbastanza banale spesso rivelano quale sistema è in uso. Molti amministratori, incautamente, non rimuovono i banner di testo presenti nei demoni e nei servizi attivi su un server: in questo caso non c'è bisogno di alcuna tecnica di fingerprinting, ma basta semplicemente enumerare le porte aperte sull'host e fare telnet su di queste aspettando il messaggio di ritorno, come mostra il seguente esempio:

```
[neo@localhost]$ telnet target.host.com
Trying 192.168.1.30...
Connected to target.host.com.
Escape character is '^]'.
SunOS 5.7
login: root
```

Dall'esempio appare evidente che abbiamo a che fare con un sistema SunOS versione 5.7, ma, purtroppo, non sempre è così semplice come in questo caso, spesso perché la porta 23 (telnet) non viene lasciata tanto grossolanamente aperta. In caso di "incontro ravvicinato" con un web server, può essere utile eseguire telnet sulla porta 80 e catturare il banner per capire con chi "abbiamo a che fare". Una volta collegati sulla porta 80, basta digitare il verbo *"GET / HTTP /1.0"* per avere un responso dal server.

```
[neo@localhost]$ telnet target.host.com 80
Trying 192.168.1.30...
Connected to target.host.com (192.168.1.30).
Escape character is '^]'.
GET / HTTP/1.0
HTTP/1.1 301 Moved Permanently
Date: Thu, 26 Jun 2003 08:15:47 GMT
Server: Apache/1.3.27 (Unix) PHP/4.2.2
Location: http://target.host.com/
Connection: close
Content-Type: text/html; charset=iso-8859-1
<!DOCTYPE HTML PUBLIC
                    "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
....
```

Nell'header di risposta, compare quasi sempre il campo *"Server:"* in cui viene indicato il web server che sta dall'altra parte: è ovvio che trovandosi di fronte ad Apache (come in questo caso), vuol dire che quasi certamente dall'altra parte è in esecuzione un sistema Linux/Unix, mentre in presenza del banner di IIS (*Internet Information Server*) sarà altrettanto ovvio concludere che

Fig. 1: Nmap è il security scanner creato da Fyodor, più conosciuto fra gli hacker. Si può scaricare liberamente dal sito *http://www.insecure.org*; è disponibile per sistemi Linux e per sistemi Windows.

Tool di OS fingerprinting

Nmap non è il solo tool per effettuare il riconoscimento di sistemi operativi da remoto. L'origine di questi programmi risale a *sirc*, scritto da Johan, che è il primo rudimentale "OS detector" prodotto, in grado di distinguere Win95, 4.4BSD o Linux dal test di alcuni flag TCP. Altri tool che hanno fatto la storia degli OS scanner, negli anni 1998-1999, sono *checkos* di Shok e SS 3.11, scritto dall'hacker su1d e capace di identificare ben 12 sistemi operativi diversi. Successivamente *Queso* e in seguito *Nmap*, introdussero il concetto esteso di "fingerprint" di un sistema operativo basato sul TCP/IP stack. Di recente un altro tool per Linux, chiamato *RINGv2*, è stato rilasciato alla comunità hacker.

Fig. 2: Il porting di Nmap su piattaforme Windows richiede la presenza delle librerie Winpcap, prelevabili dal Politecnico di Torino all'indirizzo *http://netgroup-serv.polito.it/winpcap/*.

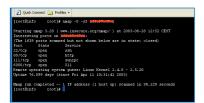



Fig. 3: **Nmap all'opera sotto Linux. Il tool rileva in pochi secondi le porte aperte e i servizi ad esse associati, identificando (parametro –O) anche il tipo di sistema operativo (Linux Kernel) grazie alle tecniche di OS fingerprinting.**

Fig. 4: **Nmap per Windows è dotato di una comoda interfaccia grafica (GUI) che consente di lanciare le scansioni specificando i parametri in maniera diretta. Prima di eseguire una scansione è necessario avviare il servizio di Nmap dalla finestra Service dell'interfaccia grafica (Start Service).**

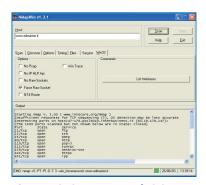



Fig. 5: **Windows XP non è del tutto compatibile con Nmap, di conseguenza è consigliabile usare l'opzione "Force RAW Socket" durante le scansioni. Tale opzione richiede i privilegi di Administrator.**

siamo in presenza di Windows 2000/XP (4.0/5.0 nel primo caso, 5.1 nel secondo). Altri banner utili si possono ricavare comunque dai servizi FTP, SSH, SMTP e POP3, rispettivamente sulle porte di accesso 21, 22, 25 e 110 (nell'esempio che segue sono mostrati alcuni esempi di connessioni molto comuni). Sotto Windows può essere invece interessante collegarsi alle porte NetBIOS (*135/139*) con un samba client per indagare sul dominio e sulla versione del LAN Manager:

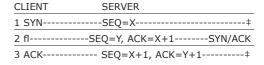
```
[neo@localhost]$ ftp target.host.com
Connected to target.host.com
220 Serv-U FTP Server v4.0 for WinSock ready...
Name (target.host.com:root): root
331 User name okay, need password.
Password:
[neo@localhost]$ ssh -v 192.168.1.30
OpenSSH_3.4p1, SSH protocols 1.5/2.0, OpenSSL
                                        0x0090609f
[neo@localhost]$ smbclient -L 192.168.1.30
added interface ip=192.168.1.30
    bcast=192.168.1.255 nmask=255.255.255.0
session request to 192.168.1.30 failed (Called
                                  name not present)
session request to 192 failed (Called name not
                                          present)
Password:
Anonymous login successful
Domain=[TARGET-DOMAIN] OS=[Windows 5.0]
        Server=[Windows 2000 LAN Manager]
    Sharename      Type      Comment
    ---------      ----      -------
Error returning browse list:
                       NT_STATUS_ACCESS_DENIED
```

ANALISI DELLE PORTE

La tecnica di analizzare i banner di risposta dei servizi attivi fornisce quasi sempre risultati interessanti, ma in molti casi gli amministratori più accorti disabilitano i banner dei servizi per rendere più difficile il compito degli hacker. In questi casi occorre ricorrere ad un'altra tecnica, dettata dal buon senso e dall'intuito, che riguarda l'analisi delle porte aperte. Lo studio delle porte aperte, e dei servizi in esecuzione, spesso rivela in maniera esatta che tipo di sistema è in esecuzione: la presenza di porte come la 23 (telnet) o la 22 (ssh) su una macchina Windows è alquanto inverosimile, così come la presenza delle porte 135/139/445 (NetBIOS) su macchine Unix è da escludere. L'esistenza o meno di determinati numeri di porta è indicativa del tipo di servizi attivi, di conseguenza servizi come SSH e Telnet segnaleranno, con molta probabilità, macchine Linux/Unix, così come il servizio NetBIOS apparterrà di sicuro ai sistemi operativi di Microsoft. Esistono molte altre porte significative, come la 1433/1434 che si riferisce al database MS-SQL Server e che identifica in maniera univoca un sistema Windows 2000/XP; allo stesso modo la 3306, relativa a MySQL, è solitamente presente su macchine Linux (anche se esiste una versione di MySQL per sistemi Win32). Un elenco completo di porte e servizi corrispondenti (*well know ports*) è disponibile su Internet nei link indicati in queste pagine. Discorso diverso va fatto per la porta 80, sulla quale, come abbiamo visto prima, può trovarsi qualsiasi tipo di web server per diversi sistemi operativi. Resta inteso che il discorso sulla corrispondenza fra *<porte, sistemi op.>* è sempre concepito in termini probabilistici, perché nulla vieta ad un host Windows di montare un server SSH sulla porta 22 oppure ad una macchina Linux di attivare il servizio Samba (protocollo che emula il NetBIOS di Microsoft) falsando quanto detto finora. Esistono comunque tecniche più avanzate rispetto all'analisi di banner e porte, che consentono di investigare a fondo sul tipo di sistema operativo in esecuzione da remoto.

OS FINGERPRINTING

Determinare un S.O. di una macchina mentre si è loggati su di essa è una cosa abbastanza semplice e non richiede particolari competenze; quello che invece è più complicato da fare, è capire il sistema in uso da remoto, senza cioè accedere alla macchina. L'OS Fingerprinting è il nome di quella tecnica in grado di determinare in maniera esatta (o con probabilità molto elevata) il tipo di sistema operativo di un certo host attraverso l'analisi di particolari informazioni e caratteristiche fornite dallo stesso host. Uno dei tool sicuramente più noti, vista la sua semplicità d'uso e il fatto che è di pubblico dominio, è senz'altro Nmap, creato dall'hacker noto col nome di *Fyodor* (*fyodor@insecure.org*). Le tecniche di fingerprinting implementate da Nmap sono diverse e sono tutte basate sull'analisi dello stack del protocollo TCP/IP. Per completezza diremo soltanto che una qualsiasi connessione TCP/IP tra due computer viene instaurata seguendo un meccanismo noto come "*3-Way Hand -Shaking*", ovvero stretta di mano "a tre vie".

```
CLIENT              SERVER
1 SYN--------------SEQ=X-------------------------‡
2 fl--------------SEQ=Y, ACK=X+1--------SYN/ACK
3 ACK------------- SEQ=X+1, ACK=Y+1----------‡
```

Il client deve sincronizzarsi col server prima di iniziare a trasmettere dati, di conseguenza manda un pacchetto di tipo *SYN*, che contiene anche un certo numero X (generato in maniera casuale) detto ISN (*Initial Sequenze Number*). Il server,

da parte sua, risponde al client con un pacchetto di tipo *SYN/ACK*, nel quale dimostra di aver ricevuto il primo pacchetto spedendo una risposta (Acknowledgement) che corrisponde al numero iniziale incrementato (*X+1*); nello stesso passo il server genera un proprio numero casuale (*Y*) e lo invia al client. La procedura si conclude quando il client risponde al server con un messaggio conclusivo di tipo *ACK* che contiene a sua volta il numero *Y* incrementato di una unità. A questo punto i due computer sono sincronizzati e possono iniziare a scambiare dati. L'RFC793 detta proprio le specifiche di questo meccanismo, che nella realtà viene implementato in maniera leggermente diversa da ciascun sistema operativo: la sostanza del protocollo non cambia tuttavia le scelte di implementazione lasciate alla libertà del costruttore (Microsoft, Sun, HP, Red Hat, Cisco) sono molteplici e sono la chiave per distinguere un sistema operativo da un altro.

TECNICHE DI FINGERPRINTING

Le principali tecniche di identificazione sono quindi basate sull'analisi dello stack TCP/IP e delle risposte fornite dall'host remoto in presenza di pacchetti "strani" o in presenza di condizioni che non sono contemplate nelle specifiche dell'RFC793, condizioni che vengono gestite in maniera diversa da ciascun sistema operativo, a seconda delle scelte progettuali fatte dai costruttori. Ecco alcuni dei principali metodi conosciuti in materia di OS fingerprinting:

- **FIN probe -** Viene spedito un pacchetto di tipo FIN (o un qualsiasi pacchetto senza il flag ACK o SYN) su una porta aperta e si attende la risposta dell'host. Le specifica dell'RFC793 sarebbe quella di non rispondere a tale pacchetto, tuttavia molte implementazioni "fuorilegge" ritornano un messaggio di RST (Windows, CISCO, HP/UX e IRIX).

- **BOGUS flag probe -** L'idea è quella di impostare un flag TCP indefinito (64 o 128) nell'header TCP di un pacchetto SYN inviato all'host. I sistemi Linux (kernel<2.0.35) mantengono tale flag inalterato anche nelle successive risposte, gli altri sistemi operativi invece resettano la connessione.

- **SYN Flooding -** Viene spedito un unico normale pacchetto di tipo SYN all'host remoto e in seguito si scarta la conseguente risposta di tipo SYN/ACK inviata dal server. Lo scarto (e il mancato invio dell'ACK finale) comporta la ritrasmissione, da parte dell'host remoto, del pacchetto SYN/ACK. Il timeout fra una trasmissione e la successiva non viene specificato nell'RFC973, quindi tali valori sono scelti dai costruttori. Misurando i tempi di ritrasmissione fra i diversi pacchetti ricevuti, si può determinare il sistema operativo in uso (tecnica usata dallo scanner RING).

- **ISN sampling -** L'idea è quella di identificare pattern e regolarità nella generazione dei numeri ISN fatta dai sistemi operativi nelle diverse implementazioni dello stack TCP. Quando infatti un host riceve un messaggio di SYN, genera in maniera pseudo-casuale un proprio numero ISN in risposta al pacchetto. Inviando diversi pacchetti SYN, è possibile calcolare di quanto differiscono fra loro i diversi numeri ISN generati dall'host remoto e verificare quanto essi siano casuali.

NMAP ALL'OPERA

Nmap si può scaricare gratuitamente dal sito *http://www.insecure.org*; è disponibile in formato RPM e in formato TAR per sistemi Linux, ma di recente è stato fatto un porting del programma anche per piattaforma Windows 2000, grazie all'uso delle librerie Winpcap. L'installazione sotto Linux è immediata, mentre sotto Windows invece bisogna prima installare Winpcap e solo successivamente installare Nmap. Sempre per Windows è stata resa disponibile una versione del programma completa di librerie Winpcap e di installer automatizzato; al termine dell'installazione bisogna riavviare il PC e ricordarsi di far partire il servizio Nmap dalla casella Service dell'interfaccia grafica. Windows XP SP1 non è del tutto compatibile con Nmap, quindi potrebbe non essere in grado di sfruttare a pieno questo tool (si consiglia di spuntare l'opzione Force Raw Socket). Per usare Nmap basta semplicemente eseguire il tool da linea di comando specificando l'host sul quale indagare (sotto forma di indirizzo IP o di hostname) e i parametri da usare:

nmap target.host.com - Scansione delle porte dell'hostname indicato

nmap 192.168.1.30 - Scansione delle porte dell'IP address indicato

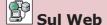
*nmap 192.168.1.** - Scansione di un'intera sottorete (192.168.1.1-255)

nmap –O 192.168.1.30 - Scansione delle porte e OS fingerprinting

Ing. Elia Florio

Fig. 6: **Chi ha visto il film "Matrix Reloaded" avrà notato la scena in cui Trinity si lancia con molta disinvoltura nell'hackeraggio di un server della centrale elettrica da disattivare. Ebbene, anche Trinity, nel film, usa Nmap (come si vede chiaramente dalla figura...).**

Fig. 7: **Particolare della scena di "Matrix Reloaded" in cui si vede chiaramente che Trinity utilizzando Nmap "V2.5BETA25" riesce a individuare la porta 22 (ssh) sul server da bucare. L'intrusione nel server da parte di Trinity avviene sfruttando un bug realmente esistito (SSHv1 CRC32) e noto alla comunità hacker.**

Sul Web

Lista completa di porte e servizi noti
http://www.neohapsis.com/neolabs/neo-ports/neo-ports.html

Sito Ufficiale di Nmap, gestito da Fyodor
www.insecure.org

RFC793 (DARPA TCP Specification)
http://www.ietf.org/rfc/rfc793.txt

Librerie Winpcap richieste da Nmap
http://winpcap.polito.it/install/default.htm

Parametri

Altri parametri utili per un uso "intelligente" di Nmap sono:

-sS - Esegue una scansione delle porte "stealth" inviando pacchetti SYN e senza connettersi
-P0 - Scansione per sistemi che non rispondono al PING (protetti da firewall)
-v - Verbose. Aumenta il livello di dettaglio nell'output prodotto da Nmap.

Crobots for ever

Torneo CRobots 2k3

Ogni anno, intorno al mese di marzo, la comunità crobotica inizia ad animarsi. Ci sono da proporre le nuove idee sugli sviluppi futuri del compilatore che poi il nostro Maurizio Camangi presenterà in forma di supplica al misterioso possessore dei sorgenti, c'è da organizzare il nuovo torneo e, ovviamente, da scrivere l'articolo di presentazione…

Nei riguardi del fortunato estensore, naturalmente, siamo molto selettivi: una competizione così importante richiede cronisti all'altezza. Il candidato ideale deve essere un po' fuori di testa, saper battere qualche lettera sulla tastiera di un PC e, necessariamente, non aver mai vinto un torneo di crobots né nutrire alcuna speranza in tal senso nemmeno per il futuro. In caso contrario, infatti, come potrebbe rimanere al di sopra delle parti e fornire un resoconto corretto e veritiero della manifestazione?
Ebbene, pare che, per una singolare concomitanza di coincidenze, io sia rimasto l'unico in Italia a soddisfare a pieno titolo l'ultimo punto dell'elenco e, conseguentemente, eccomi qua.
Naturalmente sono una persona di mondo, e non serbo il minimo rancore nei confronti di Alessandro Carlin che, al contrario, si è portato a casa ben un paio di titoli nel giro di due anni (l'ingordo!). Posso solo esprimere il desiderio che questa volta sia decisamente impegnato con le ricerche sugli sfregamenti molecolari all'università di Padova e non abbia il tempo di preparare un altro mostro iper competitivo. In caso contrario sarà opportuno cercare di squalificarlo per comportamento antisportivo(*)…
Sistemato così il campione in carica passo a fare gli auguri a un'altra colonna della nostra manifestazione: Michelangelo Messina, infatti, ha messo su famiglia. Chissà che ora non la smetta di scrivere combattenti che mandano sistematicamente in crisi i miei.
Passando alle cose serie, qualche novità profila all'orizzonte anche quest'anno. Il misterioso benefattore non è più solo: pare che esista un altro possessore dei sorgenti di crobots, tal Bill Jones, che si è fatto avanti offrendosi di soddisfare le eventuali richieste di modifiche al compilatore. Per prima cosa dovrà riuscire a sincronizzare il suo codice con crobots2000 (pare che il nuovo debugger sia stato molto apprezzato, soprattutto da Daniele Nuzzo). Poi vedremo che succederà. Personalmente troverei molto utile la presenza di una sorta di 'memoria storica' del crobot, magari da implementare sotto forma di variabili permanenti che vengano gestite dal compilatore in maniera trasparente per il programmatore. Ogni suggeriemento, comunque, andrà valutato attentamente, dal momento che la compatibilità con il passato dovrà rimanere il criterio ispiratore di ogni futura evoluzione.
Nel frattempo lo ringraziamo per la disponibilità e lo invitiamo a partecipare al concorso 2003

(*)
Anche se sono sicuro che la cosa sia stata recepita da tutti, preciso che si tratta di sfottò all'indirizzo del bicampione.

IL TORNEO DI CROBOTS 2K3… CHE È ORAMAI IMMINENTE!

La data ufficiale non è ancora stata fissata (controllate gli eventuali aggiornamenti sul sito *www.ioprogrammo.net /crobots*) ma è quasi certo che, anche quest'anno, le iscrizioni si chiuderanno alla mezzanotte del 31 ottobre 2003. Stranamente immutate le regole di partecipazione: hanno funzionato talmente bene la passata edizione che abbiamo deciso di utilizzarle anche quest'anno. Il bando ufficiale sarà reperibile, al solito, presso i siti, ufficiali e non, dedicati all'evento. Di seguito viene comunque riportato un elenco sintetico dei principali punti.

Modalità di iscrizione

- È possibile inviare fino a 4 crobots alla competizione

1. Uno dovrà rientrare al di sotto della soglia delle 500 istruzioni comprese e parteciperà al torneo dei *MicroRobot*.
2. Uno avrà a disposizione 1000 Vb di codice, e potrà essere iscritto alla categoria *ClassicBots* (in cui confluiranno anche i partecipanti alla sezione precedente).
3. Il terzo potrà sfruttare l'intero limite di indirizzamento consentito dal compilatore (2000 VirtuaByte) e disputerà il *Torneo Generale* (nel quale si sfideranno i robot di tutte e tre le classi di peso).
4. Il quarto concorrente, infine, sarà a discrezione del programmatore, che potrà scegliere in quale delle tre precedenti categorie far rientrare la sua opera.

- Il numero dei robot non è, ovviamente, vincolante:

fermo restando il limite superiore di quattro unità per autore, è possibile partecipare all'evento anche con un unico combattente.

Modalità di combattimento

- I robot andranno spediti all'indirizzo di posta elettronica *ascheri@edmaster.it* (oppure inviati con gli altri mezzi previsti nel bando).
- Tutti i robot arrivati saranno divisi in gruppi di al massimo 32 programmi (sempre nel caso, naturalmente, che il loro numero ecceda tale cifra), che disputeranno i gironi eliminatori. Questi forniranno i nomi, sia direttamente che tramite ripescaggio, dei qualificati per la fase finale.

I crobot si affronteranno, in tutte le fasi della competizione, secondo due modalità differenti:

1. Nello scontro 4vs4, naturalmente, ad ogni match partecipano quattro robot. Ogni sfidante deve incontrare ciascuno degli avversari almeno 2000 volte: il fattore di ripetizione varierà quindi a seconda dell'ampiezza del girone.
2. Nello scontro F2F, invece, ogni robot affronta a singolar tenzone ciascuno dei componenti del girone, con un fattore di ripetizione pari a 2000.

Per ridurre la durata degli incontri, viene posto il "limite temporale" dei 200.000 cicli virtuali di CPU, al termine dei quali la partita viene dichiarata patta tra i sopravvissuti.

- I punteggi saranno assegnati secondo lo schema *Pranzo*:
    - 12 punti al robot vittorioso;
    - 3 punti ai superstiti di un pareggio a due;
    - 2 punti ai superstiti di un pareggio a tre;
    - 1 punto ai superstiti di un pareggio a quattro con danni superiori al 40%;
    - 0 punti ai superstiti di un pareggio a quattro con danni inferiori al 40%;
- La classifica finale uscirà dal mixaggio tra le due graduatorie, con pesi 5*4vs4 e 1*F2F, per sottolineare la difficoltà dello scontro a 4.
- La finale sarà disputata secondo le stesse regole.

La sfida è dunque lanciata, accorrete numerosi!
Nutro la speranza che, accanto a quanti hanno risposto all'appello inoltrato in mailing list (molti dei quali sono dei nuovi arrivati) si rifaccia vivo anche quel Dario Serino che non solo dominò il torneo2k con Daryl.r, lanciando la moda dei robot che presidiano a oltranza il proprio angolo naturale, ma, da matricola, rese molto difficile la vita al vincitore nel 1999 e, da campione, difese benissimo il titolo nel 2001.
Anche quest'anno ioProgrammo, sponsor della manifestazione, mette in palio un fantastico drive Iomega Zip 750MB come premio per il vincitore della competizione e ad estrazione fra tutti i partecipanti il nuovo Iomega HDD Portable Hard Drive e lo strepitoso pacchetto McAfee Virus Scan. Quanti fossero interessati potranno ispirarsi all'enorme quantità di robot inviati dagli autori negli ultimi 13 anni, che coprono tutte le categorie in cui si articolerà la manifestazione: 500, 1000 e 2000 istruzioni. Visitate i siti dedicati a Crobots nonché il materiale presente sul CD-Rom per reperire tutto il materiale necessario allo sviluppo di un combattente agguerrito. Già che siete nei paraggi, non dimenticate di scaricare anche i vari programmi accessori che semplificano il processo di sviluppo di un automa: *Torneo XP* (per la gestione di tutti i tipi previsti di competizione), *Count 8.3* (che consente di calcolare le classifiche e effettuare analisi statistiche sui risultati), *Ecat* (necessario per una rapida 'taratura' automatica dei parametri della vostra creatura), *PPC* (preprocessore di comandi che aggiunge *#include* e *#define* al compilatore) *CRH2.0)* e il suo antagonista, *TorneoGUI* (per coordinare il lavoro dei software precedenti). Mi pare che anche per questa volta sia tutto. Dopo gli auguri di rito, non mi resta che dirvi: iscrivetevi alla mailing list ufficiale di crobos (*crobots@ioprogrammo.net*) e … fatevi sotto!!!!

Simone Ascheri,
Maurizio Camangi
Michelangelo Messina

Tips&Tricks I trucchi del mestiere

La rubrica raccoglie trucchi e piccoli pezzi di codice che solitamente non trovano posto nei manuali, ma sono frutto dell'esperienza di chi programma. Alcuni trucchi sono proposti dalla Redazione, altri provengono da una ricerca sulla Rete delle Reti, altri ancora ci giungono dai lettori. Chi vuole contribuire potrà inviarci i suoi tips&tricks preferiti che, una volta scelti, verranno pubblicati nella rubrica. Il codice completo dei tips lo trovate nel CD allegato nella directory \tips\.

VB.NET

Come "ricercare" una finestra tra quelle aperte in Windows

Questa procedura permette di "andare alla ricerca" partendo da una finestra base (per esempio quella di background di Windows il cui handle è facilmente ricavabile - attraverso l'utilizzo della API *GetDesktopWindow (Public Declare Function GetDesktopWindow Lib "user32" Alias "GetDesktopWindow" () As Long)*, di una determinata finestra o sotto-finestra avente come caption il testo indicato come parametro di funzione. In ambiente Windows XP se si cerca una finestra con caption *"Start"* si può ottenere l'handle del bottone *"START"* della barra di controllo; di conseguenza è possibile cambiare il testo in esso contenuto.

Tip fornito dal Sig. S.Tubini

```
Option explicit
Private Declare Function GetWindow Lib "user32" Alias "GetWindow"
                        (ByVal hwnd As Long, ByVal wCmd As Long) As Long
Public Declare Function GetDesktopWindow Lib "user32" () As Long
Public Declare Function lstrlen Lib "kernel32" Alias "lstrlenA"
                                         (ByVal lpString As String) As Long
Public Declare Function GetWindowTextLength Lib "user32" Alias
                    "GetWindowTextLengthA" (ByVal hWnd As Long) As Long
Private Declare Function GetWindowText Lib "user32" Alias
                     "GetWindowTextA" (ByVal hwnd As Long, ByVal lpString
                                      As String, ByVal cch As Long) As Long
Private Declare Function SetWindowText Lib "user32" Alias
                                         "SetWindowTextA" (ByVal hwnd As
Long, ByVal lpString As String) As Long
Private Const GW_CHILD = 5
Private Const GW_HWNDNEXT = 2
Public Function SearchWindow(hWnd As Long, sCaption As String) As Long
On Local Error Resume Next
Dim H1 As Long, T As String, T23 As String
 H1 = GetWindow(hWnd, GW_CHILD)
 If H1 = 0 Then Exit Function
 T = FindWindowsText(H1)
 If T = sCaption Then
    SearchWindow = H1: Exit Function
 End If
 If GetWindow(H1, GW_CHILD) > 0 Then
   SearchWindow = SearchWindow(H1, sCaption)
   If SearchWindow <> 0 Then Exit Function
 End If
    Do
     H1 = GetWindow(H1, GW_HWNDNEXT)
     If H1 = 0 Then Exit Function
     T = FindWindowsText(H1)
     If T = sCaption Then
        SearchWindow = H1: Exit Function
     End If
        If GetWindow(H1, GW_CHILD) > 0 Then
           SearchWindow = SearchWindow(H1, sCaption)
           If SearchWindow <> 0 Then Exit Function
        End If
    Loop Until H1 = 0
End Function
Public Function FindWindowsText(hWnd As Long) As String
    FindWindowsText = String$(GetWindowTextLength(hWnd) + 1, vbNull)
    GetWindowText hWnd, FindWindowsText, Len(FindWindowsText)
    FindWindowsText = Left(FindWindowsText, lstrlen(FindWindowsText))
End Function
Public Function ChangeWindowsCaption (hWnd As Long,sTxt As String)
SetWindowText hWnd, sTxt
End Function
```

Un conto alla rovescia per arrestare il sistema

Un tip per spegnere il computer dopo un determinato periodo di tempo. In caso di chiusura del programma, esso risponderà come se il tempo fosse scaduto. Per il corretto funzionamento si consiglia di inizializzare i due controlli timer con i seguenti valori di proprietà:

Timer_tempo: Intervallo 1000 / Enabled = False
Timer_controllo: Intervallo 200 / Enabled = True

Tip fornito dal Sig. M. Bruseghin

```
Private Sub Timer_tempo_Timer()
If lblTempos.Caption = 0 And lblTempom.Caption =
                              0 And lblTempoh.Caption = 0 Then
Timer_tempo.Enabled = False
Else
lblTempos.Caption = lblTempos.Caption - 1
End If
If lblTempos.Caption < 0 Then
lblTempom.Caption = lblTempom.Caption - 1
lblTempos.Caption = 59
End If
If lblTempom.Caption < 0 Then
lblTempoh.Caption = lblTempoh.Caption - 1
lblTempom.Caption = 59
End If
If lblTempoh.Caption < 0 Then
lblTempoh.Caption = 0
```

```
lblTempom.Caption = 59
End If
If lblTempoh.Caption = 0 Then
lblTempoh.Caption = 0
End If
If lblTempos.Caption = 0 Then
If lblTempom.Caption = 0 Then
If lblTempoh.Caption = 0 Then
Timer_tempo.Enabled = False
Form1.Show
MsgBox "Tempo scaduto!", vbOKOnly + vbExclamation, "Arresto sistema
in corso..."
` per Windows 98
Shell ("C:\WINDOWS\RUNDLL.EXE user.exe,exitwindows")
`per Windows XP
Shell ("C:\WINDOWS\RUNDLL32.EXE user,exitwindows")
End
End If
End If
End If
End Sub
Private Sub Timer_controllo_Timer()
If lblTempos.Caption = 0 And lblTempom.Caption =
                    0 And lblTempoh.Caption = 0 Then
cmdAccendi.Enabled = False
Else
If Timer_tempo.Enabled = False Then
cmdAccendi.Enabled = True
End If
End If
End Sub
`codice dei vari CommandButton
Private Sub cmdAccendi_Click()
Timer_tempo.Enabled = True
cmdAccendi.Enabled = False
End Sub
Private Sub cmdStop_Click()
Timer_tempo.Enabled = False
If lblTempoh.Caption = 0 And lblTempom.Caption = 0 And
lblTempos.Caption = 0 Then
cmdAccendi.Enabled = False
Else
cmdAccendi.Enabled = True
End If
End Sub
Private Sub cmdImpostaTime_Click()
If Timer_tempo.Enabled = False Then
cmdAccendi.Enabled = False
End If
If txtOra.Text > "24" Then
MsgBox "Il Timer dell'ora non è accettabile", vbOKOnly + vbCritical,
                                                  "Errore MB
TimeOut! 1.0"
txtOra.Text = ""
txtOra.SetFocus
Else
lblTempoh.Caption = txtOra.Text
End If
If txtMinuti.Text > "59" Then
MsgBox "Il Timer dei minuti non è accettabile", vbOKOnly + vbCritical,
                                                  "Errore MB
TimeOut! 1.0"
txtMinuti.Text = ""
txtMinuti.SetFocus
Else
lblTempom.Caption = txtMinuti.Text
End If
If txtSecondi.Text > "59" Then
MsgBox "Il Timer dei secondi non è accettabile", vbOKOnly +
                                          vbCritical, "Errore MB
TimeOut! 1.0"
txtSecondi.Text = ""
txtSecondi.SetFocus
Else
lblTempos.Caption = txtSecondi.Text
End If
If txtOra.Text = "" Then
lblTempoh.Caption = 0
End If
If txtMinuti.Text = "" Then
lblTempom.Caption = 0
End If
If txtSecondi.Text = "" Then
lblTempos.Caption = 0
End If
End Sub
Private Sub optImpostazioni_Click()
txtOra.Enabled = False
txtMinuti.Enabled = False
txtSecondi.Enabled = False
cmdImpostaTime.Enabled = False
cmdStop.Enabled = False
End Sub
Private Sub Form_Unload(Cancel As Integer)
MsgBox "Chiusura non consentita", vbOKOnly + vbCritical,
                                  "Arresto sistema in corso..."
' per Windows 98
Shell ("C:\WINDOWS\RUNDLL.EXE user.exe,exitwindows")
'per Windows XP
Shell ("C:\WINDOWS\RUNDLL32.EXE user,exitwindows")
End
End Sub
```

Come copiare una lista di IP in un vettore

Spesso su internet si trovano lunghissime liste di IP (Internet Protocol). Volendo creare un programma per gestire tali indirizzi è comodo averli sotto forma di vettori. Il tip proposto prende in considerazione gli ip nella forma: 255.255.255.255:8080 prelevati da un controllo textbox e poi spezzettati in due array semplici: *ip()* e *port()*
Tip fornito dal Sig. S. Rinaldo

```
Private Sub Command1_Click()
  Dim temp() As String
  Dim ip()
  Dim port()
  Dim fine As Integer
```

```
    Dim i, j As Integer
    temp = Split(txtIP.Text, vbCrLf)
    j = -1
        For i = LBound(temp) To UBound(temp)
         fine = InStr(1, temp(i), ":")
         If ((fine > 7) And (fine < 17)) Then
             j = j + 1
             ReDim Preserve ip(j)
             ReDim Preserve port(j)
             ip(i) = Mid(temp(i), 1, fine - 1)
             port(i) = Mid(temp(i), fine + 1, Len(temp(i)))
         End If
    Next i
End Sub
```

Un semplice Server proxy

Una semplice applicazione che funge da Server Proxy consentendo la condivisione e l'accesso ai documenti ipertestuali del WEB tra tutti i computer di una rete locale. Per poterlo utilizzare, all'interno del browser digitare: "*http://IP_PROXY:8080/IP-SERVER/pagina.htm*

Tip fornito dal Sig. M.Nicosia

```
Option Explicit
Dim i As Integer
Private Sub delay(interval As Single)
Dim s As Single
s = Timer
Do While Timer < (s + interval)
    DoEvents
Loop
End Sub
Private Sub parse(buffer As String, ByRef server As String,
                                        ByRef richiesta As String)
On Error Resume Next
Dim url As String
url = Left$(buffer, InStr(buffer, "HTTP/1.1") - 2)
url = Right$(url, Len(url) - 5)
If InStr(url, "/") = 0 Then
    server = url
    richiesta = "GET /"
Else
    server = Left$(url, InStr(url, "/") - 1)
    richiesta = "GET " & Right$(url, Len(url) - Len(server)) & vbCrLf
End If
End Sub
Private Sub Form_Load()
wskserver.LocalPort = 8080
wskserver.Listen
i = 0
End Sub
Private Sub wskconnect_DataArrival(Index As Integer, ByVal
                                               bytesTotal As Long)
On Error Resume Next
Dim buffer As String
Dim server As String
Dim richiesta As String
wskconnect(Index).GetData buffer
parse buffer, server, richiesta
Load wskweb(Index)
wskweb(Index).Connect server, 80
delay 1
```

IL TIP-ONE del mese

Convertire un filmato flash in EXE

Il tip consente di convertire un qualunque filmato formato Macromedia Flash in un comune file eseguibile di Windows, ovvero un .EXE. Si tratta di una funzione che accetta tre argomenti:il file sorgente (file formato Flash), il file di destinazione (file .exe) e il percorso del Flash Player installato nel proprio sistema.

Tips fornito dal sig. G. Cassano

```
function Swf2Exe(S, D, F: string): string;
var
 SourceStream, DestinyStream, LinkStream: TFileStream;
 flag: Cardinal;
 SwfFileSize: integer;
begin
 result := 'something error';
 DestinyStream := TFileStream.Create(D, fmCreate);
 try
   LinkStream := TFileStream.Create(F, fmOpenRead or
                                          fmShareExclusive);
   try
     DestinyStream.CopyFrom(LinkStream, 0);
   finally
     LinkStream.Free;
   end;

   SourceStream := TFileStream.Create(S, fmOpenRead or
                                          fmShareExclusive);
   try
     DestinyStream.CopyFrom(SourceStream, 0);
     flag := $FA123456;
     DestinyStream.WriteBuffer(flag, sizeof(integer));
     SwfFileSize := SourceStream.Size;
     DestinyStream.WriteBuffer(SwfFileSize, sizeof(integer));
     result := '';
   finally
     SourceStream.Free;
   end;
 finally
   DestinyStream.Free;
 end;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
 Swf2Exe('c:\prova.swf', 'c:\provai.exe', 'D:\FlashPla.exe');
end;
```

```
wskweb(Index).SendData richiesta
End Sub
Private Sub wskconnect_SendComplete(Index As Integer)
wskconnect(Index).Close
wskweb(Index).Close
Unload wskconnect(Index)
Unload wskweb(Index)
End Sub
Private Sub wskserver_ConnectionRequest(ByVal requestID As Long)
    i = i + 1
    Load wskconnect(i)
    wskconnect(i).Accept requestID
    delay 1
End Sub
Private Sub wskweb_DataArrival(Index As Integer, ByVal bytesTotal As Long)
    On Error Resume Next
    Dim risposta As String
    wskweb(Index).GetData risposta
    wskconnect(Index).SendData risposta
End Sub
```

Come calcolare il crc32 di alcune tipologie di dati

La semplice applicazione Visual Basic permette di calcolare il CRC32 di una stringa, di una array di byte o di un file. Ricordiamo che il controllo del CRC32 consente, mediante una tecnica di codifica dei bit, di ottenere un elevato grado di protezione del messaggio in oggetto, grazie alla rilevazione di un'alta percentuale d'errori.
Tip fornito dal Sig. S.Tubini

```
Option Explicit
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory"
            (Destination As Any, Source As Any, ByVal Length As Long)
Private crc32table(255) As Long
Private Crc32TableSet As Boolean
Public Sub CreateTable(Optional lPolinomio As Long = &HEDB88320)
  Dim I      As Long
  Dim j      As Long
  Dim lCrc   As Long
  For I = 1 To 255 Step 1
    lCrc = I
    j = 8
    For j = 1 To 8 Step 1
      If (lCrc And 1) Then
        lCrc = ((lCrc And &HFFFFFFFE) \ 2&) And &H7FFFFFFF
        lCrc = lCrc Xor lPolinomio
      Else
        lCrc = ((lCrc And &HFFFFFFFE) \ 2&) And &H7FFFFFFF
      End If
    Next
    crc32table(I) = lCrc
Next I
Crc32TableSet = True
End Sub
Public Function CalcCRC32(ByteArr() As Byte) As Long
  Dim I As Long
  Dim CRC32Val As Long
  Dim ArrLen As Long
  Dim LongBytes(3) As Byte
  If (Not Crc32TableSet) Then CreateTable
  CRC32Val = -1
  ArrLen = UBound(ByteArr())
  For I = 0 To ArrLen Step 1
      SplitLongValues CRC32Val, LongBytes(): LongBytes(3) = 0
      CRC32Val = crc32table((CRC32Val Xor ByteArr(I)) And &HFF)
      CRC32Val = CRC32Val Xor MergeLongValues(LongBytes())
  Next
  CalcCRC32 = CRC32Val
End Function
Public Function CalcCRC32FromString(sStr As String) As Long
Dim bArr() As Byte, I As Long
  If Len(sStr) > 0 Then
    ReDim bArr(0 To (Len(sStr) - 1)) As Byte
    For I = 1 To Len(sStr) Step 1
      bArr(I - 1) = Asc(Mid(sStr, I, 1))
    Next
     CalcCRC32FromString = CalcCRC32(bArr())
  End If
End Function
Public Function CalcCRC32FromFile(sFile As String) As Long
On Local Error Resume Next
Dim bArr() As Byte, I As Long, L As Long
  If FileLen(sFile) > 0 Then
    If Err <> 0 Then Err = 0: Exit Function
    ReDim bArr(0 To (FileLen(sFile) - 1)) As Byte
    L = FreeFile()
    Open sFile For Binary As L
      Get L, , bArr()
    Close L
    CalcCRC32FromFile = CalcCRC32(bArr())
  End If
End Function
Public Sub SplitLongValues(lValue As Long, ByteArr() As Byte)
  CopyMemory ByteArr(0), lValue, 4
End Sub
Public Sub SplitIntegerValues(iValue As Integer, ByteArr() As Byte)
  CopyMemory ByteArr(0), lValue, 2
End Sub
Public Function MergeLongValues(ByteArr() As Byte) As Long
  CopyMemory MergeLongValues, ByteArr(0), 4
End Function
Public Function MergeIntegerValues(ByteArr() As Byte) As Integer
  CopyMemory MergeIntegerValues, ByteArr(0), 2
End Function
```

Java Script

Come creare una directory

L'argomento è la creazione di una directory con la classe *File* contenuta nel package *java.io*. Per creare una cartella del tipo: *c:\programmi\java\esempio*. Si può usare il metodo *mkdir()* della classe *File* suddetta, ma, ahimè, tale metodo non permette di creare la cartella *"esempio"* se la cartella Java non è stata creata precedentemente e la stessa considerazione vale per la cartella padre *"programmi"*. Una semplice soluzione può essere quella di utilizzare lo *StringTokenizer*

per creare ricorsivamente le sottocartelle.
Tip fornito dal Sig. M. Barbaro

```
StringTokenizer tokenDirectory = new StringTokenizer(
                                          percorsoCompleto, "\\");
    StringBuffer stringaSottoDirectory = new StringBuffer("");
    while(tokenDirectory.hasMoreElements()) {
        stringaSottoDirectory.append(tokenDirectory.nextToken())
                                                      .append("\\");
        File sottoDirectory = new File(stringaSottoDirectory.toString());
        if(!sottoDirectory.exists()) {sottoDirectory.mkdir(); }
}
```

Come controllare la porta parallela da qualunque sistema Operativo

Il controllo delle porte I/O è un problema delicato, ove il software e l'hardware arrivano a sfiorarsi. È possibile inviare al PC istruzioni maldestre che compromettono il funzionamento della macchina. Per scongiurare eventuali problemi, i sistemi operativi della Microsoft dopo NT4 hanno preso il controllo completo delle porte I/O. Le istruzioni di lettura e scrittura come *port* nel PASCAL e *inp* nel C generano un errore istruzione privilegiata terminando il flusso del programma. Per questo motivo negli articoli di elettronica applicata pubblicati da questa rivista si specifica si utilizzare come sistema operativo Windows 98 o 95. Il problema si risolve utilizzando un driver. Ad esempio si può sfruttare *PortTalk Driver for Windows NT/2000/XP* downlodabile da *http:// www.beyondlogic.org*. Questo driver funziona correttamente con i sistemi Windows NT/2000/XP (in allegato trovate il file porttalk22.zip con la documentazione ufficiale). Rimane il problema di non conoscere a priori il sistema operativo della macchina.

Come sospendere l'esecuzione di un programma

A volte è necessario sospendere l'esecuzione di un'applicazione per un determinato periodo di tempo. La funzione "pausa" scritta in C+, ed oggetto di questo tip, fa proprio al caso nostro: sospende, infatti, l'esecuzione di un programma per un intervallo di tempo espresso in secondi. Di seguito viene riportato il listato corredato da un esempio di utilizzo.
Tip fornito dal Sig. R.Pizzolante

```
#include <iostream.h>
#include <time.h>
/*funzione pausa*/
void pausa(double sec) {
   clock_t t = (clock_t)(clock() + (CLOCKS_PER_SEC * sec));
   while (clock() < t);
   return; }
/*Esempio di utilizzo :
stampa la stringa puntata da s con effetto "macchina da scrivere" */
int main(void) {
   char *s = "Raffaele Pizzolante";
   for ( ; *s; s++)    {
      cout << *s;
      pausa(1.0 / 5.0); /*1/5 di secondo*/    }
   return 0; }
```

Come monitorare i processi di Windows

Questo tip mostra come sia possibile monitorare i processi di Windows con un'applicazione C#. In particolare si possono ottenere dettagliate informazioni riguardanti l'ID del processo, la memoria utilizzata, il modulo principale, etc.; La classe *"Process"* del Namespace *"System.Diagnostic"* consente, inoltre, di inviare un segnale *"Kill"* ai processi attivi.
Tip fornito dal Sig D.Camassa

```
private void startMonitor() {
   while(true) {
      //Per l'aggiornamento dell'albero è necessario utilizzare un delegato
      ProcessTree.BeginInvoke(new UpdateTree(UpdateTreeDelegate));
      //Refresh ogni 10 secondi
      thProcess.Join(5000);} }
private void UpdateTreeDelegate(){
   ProcessTree.Nodes[0].Nodes.Clear();
   ProcessTree.BeginUpdate();
   //ottengo un array dei processi attivi
   process=System.Diagnostics.Process.GetProcesses();
   //costruisco l'albero dei processi
   for(int i=0;i<process.Length;i++)   {
      ProcessTree.Nodes[0].Nodes.Add(new
      TreeNode(process[i].ProcessName,1,1));   }
   ProcessTree.EndUpdate();
   //Riseleziono l'ultimo nodo scelto prima del refresh
   ProcessTree.SelectedNode=ProcessTree.Nodes[0].Nodes[
                                             SelectedNodeIndex];}
private void ProcessTree_AfterSelect(object sender,
                        System.Windows.Forms.TreeViewEventArgs e) {
   if(!(((TreeView)sender).SelectedNode.Text.Equals("Processi")))   {
      try      {
         SelectedNodeIndex=ProcessTree.SelectedNode.Index;
         Process pr=Process.GetProcessById(process[((TreeView)sender).
         SelectedNode.Index].Id);
         StringBuilder sb=new StringBuilder();
         //Ottengo tutte le informazioni sul processo selezionato
         sb.Append("Name:"+pr.ProcessName+"\n");
         sb.Append("ID:"+pr.Id+"\n");
         sb.Append("Priorità:"+pr.BasePriority+"\n");
         sb.Append("Handle:"+pr.Handle +"\n");
         sb.Append("Modulo principale:"+pr.MainModule+"\n");
         sb.Append("Dimensione della memoria
         privata:"+pr.PrivateMemorySize +"\n");
         sb.Append("Avvio:"+pr.StartTime +"\n");
         sb.Append("thread:"+pr.Threads+"\n");
         sb.Append("Tempo utente di processore:"+
                                       pr.UserProcessorTime +"\n");
```

```
        sb.Append("Memoria virtuale usata:"+
                                          pr.VirtualMemorySize+"\n");
        InformationProcess.Text=sb.ToString();   }
      catch(Exception err)    {
        InformationProcess.Text="Errore nella lettura delle informazioni:
                                                    "+err.Message; }
    }
}
private void menuItem1_Click(object sender, System.EventArgs e) {
  //ATTENZIONE:Uccido il processo selezionato
  try   {
    process[ProcessTree.SelectedNode.Index].Kill();
    ProcessTree.SelectedNode=ProcessTree.Nodes[0].Nodes[0];    }
  catch(Exception er){MessageBox.Show("Errore: "+er.Message);
  }
}
```

Riconoscere il sistema operativo

Per risolvere questo problema si è definita una classe *Cport* che, oltre incorporare il codice di gestione del driver va a riconoscere il sistema operativo. Questa operazione è svolta dalla seguente funzione

```
BOOL CPort::GetOpSystem() {
  OSVERSIONINFO *osvi;
  osvi = new(OSVERSIONINFO);
  osvi->dwOSVersionInfoSize=sizeof(OSVERSIONINFO) ;
  GetVersionEx (osvi);
  bIsWindowsNTorLater = (osvi->dwPlatformId ==
                                     VER_PLATFORM_WIN32_NT );
  try {delete osvi;}
  catch(...){;};
   return bIsWindowsNTorLater;
}
```

per cui prima di leggere/scrivere il valore della porta si controlla il tipo di sistema operativo:

```
unsigned char CPort::inport(unsigned short PortAddress) {
   if (!bIsWindowsNTorLater) { return _inp(PortAddress);
   };
   unsigned int error;
   DWORD BytesReturned;
   unsigned char Buffer[3];
   unsigned short * pBuffer;
   pBuffer = (unsigned short *)&Buffer;
   *pBuffer = PortAddress;
   error = DeviceIoControl(PortTalk_Handle, IOCTL_READ_PORT_UCHAR,
                        &Buffer, 2, &Buffer, 1, &BytesReturned,NULL);
   if (!error)
     AfxMessageBox("Errore di comunicazione con il driver",MB_OK,0);
   return(Buffer[0]); }
void CPort::outport(unsigned short PortAddress, unsigned char byte)
{
   if (!bIsWindowsNTorLater) {
     _outp(PortAddress,byte);
   return; };
   unsigned int error;
   DWORD BytesReturned;
   unsigned char Buffer[3];
   unsigned short * pBuffer;
   pBuffer = (unsigned short *)&Buffer[0];
   *pBuffer = PortAddress;
   Buffer[2] = byte;
   error = DeviceIoControl(PortTalk_Handle,IOCTL_WRITE_PORT_UCHAR,
                          &Buffer, 3, NULL, 0, &BytesReturned,NULL);
   if (!error)
     AfxMessageBox("Errore di comunicazione con il driver",MB_OK,0);
}
```

L'utilizzo della classe è molto semplice. Bisogna dichiarare una variabile di tipo *Cport* e assegnare il corretto valore all'indirizzo. A questo punto l'operazione di lettura è svolta attraverso il metodo *leggi* che aggiorna le proprietà value (valore decimale della variabile), e *bit[i]* (array contenente i singoli bit). L'operazione di scrittura avviene sul singolo bit attraverso il metodo *setout(unsigned int value, unsigned char bitnum)*. Per funzionare correttamente insieme alla calsse *Cport* è necessario includere nella cartella dell'eseguibile anche il file *porttalk.sys* (il vero driver). Tutto così semplice? No. Installare un servizio o un driver in Windows NT o 2000 o XP non è cosa da tutti! O meglio bisogna avere i diritti di amministratori. Questo vale solo per la prima volta. Basta allora far eseguire una volta il programma dall'amministratore e poi avete il controllo delle porte I/O. In allegato trovate un progetto per il controllo della porta parallela equivalente a quello pubblicato su ioProgrammo n.57-58, ma funzionante con tutti i sistemi operativi.

Tip fornito dal Sig. F. Gobbi

Elettronica

☑ Porta parallela e infrarossi (parte seconda)

Un telecomando per pilotare il PC

In questa seconda parte realizzeremo un ricevitore a infrarossi e scopriremo come ricevere i segnali di un telecomando attraverso la porta parallela.

File sul Web
www.ioprogrammo.net/files/72/Elettronica.zip

Infrarossi e wireless

I moderni telefoni cellulari ed alcuni PC portatili sfruttano gli infrarossi per stabilire una veloce e facile (anche se lenta) connessione wireless, su internet è possibile trovare software e molti schemi per la costruzione di una interfaccia a infrarossi standard.

Il PC è una macchina dalle molteplici funzionalità e dai più disparati utilizzi, quando ci si addentra poi nel campo della programmazione ci si accorge del fatto che non esistono limiti a quello che il software può fare, se non quelli legati all'hardware di cui disponiamo. In effetti gli unici limiti del software che possiamo scrivere, sono quelli imposti dalle caratteristiche fisiche delle macchine per le quali quel software è progettato. Per mettere in pratica il progetto che ci proponiamo di realizzare, ricevere ed interpretare i segnali di un comune telecomando a infrarossi, abbiamo bisogno, prima di tutto, di estendere i limiti del nostro PC. Nel caso specifico necessitiamo di una periferica in grado di permettergli di "vedere" gli infrarossi, estendendo così i suoi "sensi". In questo articolo tratteremo la realizzazione di questa periferica sia da un punto di vista hardware (come vedremo la sua costruzione non richiede particolari abilità e conoscenze) sia dal punto di vista software, approfondendo gli aspetti più squisitamente implementativi e i possibili ulteriori sviluppi. Il mese scorso abbiamo trattato dettagliatamente gli aspetti riguardanti le problematiche di interfacciamento di dispositivi tramite le "porte" che i personal computer ci mettono a disposizione (porta parallela, porta seriale, porta joystick e USB), focalizzando la nostra attenzione su quella che, di fatto, è la più versatile e allo stesso tempo la più semplice da gestire, la porta parallela. Abbiamo così preparato le basi sulle quali si fonderà il software che ci apprestiamo a scrivere, basi che integreremo, più avanti in questo articolo, con delle nozioni di programmazione multithread necessarie a comprendere alcuni elementi di questo progetto. Prima di seguire lo sviluppo del programma dobbiamo capire che tipo di segnale ci apprestiamo a ricevere/riconoscere tramite la nostra periferica. Vediamo, quindi, come funziona un tipico telecomando a infrarossi per televisore.

COME FUNZIONA IL TELECOMANDO

Ogniqualvolta premiamo un tasto del telecomando del nostro televisore, un circuito genera un "segnale" che il ricevitore del televisore è in grado di interpretare per eseguire l'operazione desiderata. Il segnale trasmesso, che è ovviamente diverso per ogni tasto, non è altro che una stringa di "0" e "1" di una data lunghezza, in cui i valori "0" ed "1" durano un determinato lasso di tempo. Immaginate un'onda quadra, come quella in Fig. 3, nella quale ogni stato, "0" o "1", dura qualche millisecondo. Quando teniamo premuto un tasto del telecomando il segnale corrispondente viene ripetuto continuamente interponendo un tempo di pausa (di solito paragonabile alla lunghezza totale del segnale), tra un segnale e l'altro. Ciò che rende ogni tasto riconoscibile attraverso il suo segnale sono, in genere, i diversi tempi di permanenza degli stati ("0" o "1"); tuttavia la lunghezza del segnale è una caratteristica costante per ogni tasto, particolarità questa che rende notevolmente più semplice il riconoscimento. Da notare che anche se il tasto del telecomando viene rilasciato proprio nel bel mezzo della trasmissione di un segnale, quest'ultimo viene comunque terminato. In realtà quello di cui abbiamo parlato è il segnale già decodificato da un "ricevitore di infrarossi", un componente elettronico installato nel televisore e che useremo anche per il nostro circuito. Il segnale trasmesso dal telecomando è un'onda più complessa e con una frequenza maggiore, creata appositamente per evitare disturbi e cattive interpretazioni di luce infrarossa proveniente da altre fonti come le lampade ad incandescenza. Fortunatamente lavoreremo con il segnale già decodificato e reso "digitale" prima dell'ingresso nella porta parallela. A differenza di ciò che avviene in un comune televisore, il nostro software, una volta effettuata una breve fase di apprendimento, sarà potenzialmente in grado di ricevere e interpretare i segnali provenienti da un qualunque telecomando… Possiamo adesso capire come può essere riconosciuto il segnale corrispondente ad un tasto. Bisogna prima di tutto "campionarne" il segnale, misurando il tempo che intercorre tra ogni cambiamento di stato dell'onda

quadra e poi confrontarlo con un archivio che contiene questi dati per ogni tasto del telecomando. A tale scopo installeremo una interrupt service routine (ISR), una funzione che viene richiamata ogniqualvolta cambia lo stato di un pin di ingresso della porta parallela e memorizza il tempo intercorso dall'ultimo cambiamento. Vedremo più avanti come alcuni fattori, quali la velocità del nostro PC, il numero di programmi in esecuzione, quindi la percentuale di CPU impegnata e l'ottimizzazione del nostro programma, possono incidere significativamente sulla bontà del campionamento che effettuiamo e, quindi, sul riconoscimento del tasto stesso. Vediamo, a questo punto, cosa occorre per costruire il circuito e come montarlo.

COSTRUIAMO LA NOSTRA PERIFERICA

La periferica necessaria affinché il nostro PC sia in grado di "vedere" gli infrarossi è composta essenzialmente da 3 componenti tutti facili da reperire e dal costo davvero irrisorio. Ci servono: un connettore per porta parallela (tecnicamente un *connettore maschio D25*), una *resistenza da 470ohm* e un ricevitore integrato di infrarossi che lavori alle frequenze dei normali telecomandi per televisore *(36-38 KHz)*, quello usato in questo esempio è un *Siemens SFH-507-38*. Il ricevitore integrato di infrarossi è il componente scuro con tre piedini che potete vedere sia in Fig. 1 che, schematicamente, in Fig. 2. Due dei tre piedini di questo componente (quelli più vicini tra loro) servono ad alimentarlo; tensione e corrente di alimentazione sono compatibili con quelle fornite in output dalla porta parallela. Prenderemo la massa da uno dei piedini di massa della porta *(Pin 25)* e l'alimentazione direttamente dal primo piedino di output *(Pin 2)*, così facendo non dobbiamo dimenticare di porre lo stato di questo piedino ad "alto" tramite il software, diversamente il circuito non risulterà alimentato. Inoltre, collegheremo la resistenza ai capi di uno di questi piedini in modo da limitare la corrente assorbita e proteggere la porta (questa è soltanto una precauzione). Il terzo piedino del ricevitore integrato di infrarossi è l'output. Tra questo piedino e la massa varia la tensione da livello "0" a "1", rispettivamente rappresentati da

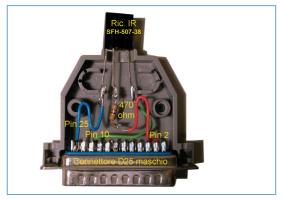



Fig. 1: **L'interno della periferica, come si può notare è interamente contenuta nel connettore.**

0volt(massa) e *+5volt*, mentre si riceve un segnale. Questi valori (come abbiamo visto nell'articolo precedente) sono direttamente compatibili con gli ingressi della porta parallela, il piedino può essere collegato, quindi, direttamente ad uno dei pin di ingresso della porta parallela. Nel nostro caso lo collegheremo al pin 10, l'unico ingresso in grado di segnalare un interrupt al sistema. Come è possibile vedere in Fig. 1, l'intero circuito può essere montato all'interno del connettore stesso, lasciando esternamente il ricevitore integrato di infrarossi, al quale deve essere data la possibilità di "vedere" fisicamente il trasmettitore del telecomando. L'esperienza comune ci suggerisce che non è necessario puntare il telecomando esattamente verso il ricevitore, questo è dovuto al fatto che i raggi infrarossi, come avviene per la luce visibile, rimbalzano sulle superfici. Una volta costruito, il circuito può essere collegato direttamente alla porta parallela, il mio consiglio è quello di farlo attraverso un cavo di prolunga in modo che il ricevitore non stia nascosto dietro il PC diventando, quindi, più difficile da raggiungere.

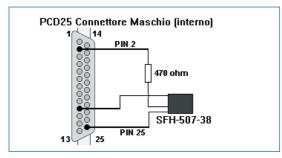



Fig. 2: **Schema della periferica.**

La nostra periferica è adesso pronta, come potete vedere non è nulla di particolarmente complesso, questo grazie anche al fatto che è possibile alimentare direttamente i componenti dalla porta parallela. L'introduzione di un circuito di alimentazione avrebbe, infatti, introdotto una certa complessità circuitale richiedendo anche qualche altro componente a protezione della porta. Per conoscere tutte le caratteristiche tecniche, lo schema equivalente e i dettagli sul funzionamento del ricevitore integrato di infrarossi è sufficiente cercare il datasheet (foglio delle caratteristiche) su internet.

INTERFACCIAMO LA PERIFERICA

Come già anticipato, la porta parallela fornisce una linea di interrupt che può segnalare i cambi di livello che si presentano sul PIN 10 della porta, Windows fornisce, però, uno strato di astrazione dall'hardware per garantirne l'accesso in maniera sicura e da parte di più utenti, per questo motivo la lettura/scrittura sulle porte e la gestione dell'interrupt non possono essere gestiti direttamente, ma devono essere effettuati tramite un driver. Su internet è possibile trovare numerosi driver freeware o shareware, il driver che usere-

Evitare i disturbi

Quando si lavora con gli infrarossi spesso si usano dei filtri di "plastica" scura per filtrare "luci" indesiderate che potrebbero alterare il campionamento, basta guardare un comunissimo televisore per notare questo accorgimento.

Elettronica

Un telecomando *per pilotare il PC*

mo per il nostro progetto è *"Universal Parallel Port"*; è shareware, supporta l'interrupt ed è prodotto da una compagnia tedesca, la *TheSycon Systemsoftware & Consulting GmbH*. Può essere scaricato gratuitamente in versione demo dal sito *http://www.thesycon.de* così come altri driver che gestiscono, tra le altre, anche la porta USB. *Universal Parallel Port*, come quasi tutto il software shareware, ha delle limitazioni nell'utilizzo, in questo caso la limitazione sta nel fatto che ogni 60 minuti il driver smette di funzionare ed è necessario riavviare il sistema.

Fig. 3: **Uno screenshot del programma dopo un campionamento.**

LA NOSTRA APPLICAZIONE

Adesso che abbiamo tutto il necessario non ci resta che scrivere la nostra applicazione, in realtà non ci cimenteremo nella scrittura di un software complesso, ma ne vedremo comunque gli aspetti salienti. Riconoscere un tasto ha senso solo se, avvenuto il riconoscimento, ne scaturisce un'azione. Nel nostro programma lasceremo questo aspetto alle vostre idee e ci occuperemo, invece, di ricevere e visualizzare i segnali del telecomando a video come una sorta di oscilloscopio. Ma veniamo ai dettagli dell'applicazione: una volta impostata l'applicazione base con il wizard di Microsoft Visual Studio, è stata creata una classe *CParallelIO* che identifica il dispositivo *"porta parallela"*. Di questa classe può essere creata una sola istanza *(singleton)* dinamica; essendo sia costruttore che distruttore privati, l'unico modo per creare/distruggere un'istanza della classe è chiamare i metodi statici *AddRef/DelRef* che tengono anche un reference counter per distruggere l'unica istanza della classe quando nessun utente la sta più utilizzando. Il costruttore della classe *CParallelIO* prova ad "aprire" il driver, attraverso la funzione *UppOpenDriver* (l'accesso alle funzioni del driver viene garantito dal file di intestazione *upp_api.h* che si trova nella directory *Source/Inc* del driver installato sotto *C:\Programmi*), se l'apertura del driver va a buon fine prova ad inizializzare la porta numero 1, quella installata sulla scheda madre. Nel caso si voglia specificare un'altra porta, ad esempio una PCI o ISA installata separatamente è sufficiente cambiare il numero della porta nella funzione *init*. L'inizializzazione della porta avviene tramite la funzione *UppInitialize*, alla quale viene specificato, tramite una costante, anche l'intenzione di utilizzare l'interrupt. Con *Universal Parallel Port* l'interrupt viene gestito sincronizzando un oggetto evento con uno creato internamente dal driver e quindi creando un thread che rimane continuamente in attesa che questo evento venga "segnalato" per poi chiamare la routine necessaria a "gestire" l'interrupt. Segue quindi nella routine *init* il codice necessario ad ottenere questi passaggi. Bisogna dire che usare l'interrupt introduce una certa complessità nel progetto che ci ripagherà in termini di pulizia del segnale ricevuto. Si potrebbe, infatti, ricevere il segnale anche interrogando continuamente la porta (*polling*), avendo, tuttavia, un notevole impatto prestazionale e, in più, con il rischio di non rilevare un cambiamento di stato, nel caso in cui questo sia troppo rapido paragonato alla nostra velocità di interrogazione della porta. Questo rischio cresce insieme alla percentuale di CPU impegnata in altri compiti. L'uso dell'interrupt invece garantisce un'occupazione di tempo macchina irrisoria, di fatto il PC verrà impegnato solo ed esclusivamente quando si riceve un segnale. Infine, la classe *CParallelIO* presenta i due metodi *EnableIRQ* e *DisableIRQ*. *EnableIRQ*, come suggerisce lo stesso nome servono ad abilitare l'interrupt.

```
BOOL CParallelIO::EnableIRQ(tISRfunc pISRfunc,
                                        void *pISRparam)
{
  if (!openPort()) return FALSE;
  // Conserva il puntatore alla funzione di callback e il
                                              suo parametro
  m_ISRfunc = pISRfunc;
  m_ISRparam= pISRparam;
  // Abilita il bit "IRQ enable" lasciando invariati tutti gli altri
  unsigned char value;
  ReadByte(OUTPORT1_OFS, &value);
  WriteByte(OUTPORT1_OFS, value |
                                  OUTPORT1_IRQENABLE);
  return TRUE;
}
BOOL CParallelIO::DisableIRQ()
{
  if (!openPort()) return FALSE;
  // Disabilita il bit "IRQ enable" lasciando invariati
                                               tutti gli altri
  unsigned char value;
  ReadByte(OUTPORT1_OFS, &value);
  WriteByte(OUTPORT1_OFS, value &
                                  ~OUTPORT1_IRQENABLE);
  return TRUE;
}
```

Come già anticipato, per abilitare questa funzione della porta bisogna accedere al terzo registro della porta parallela ed abilitare il bit 4, in questo modo viene segnalato all'hardware che l'interrupt è stato abilitato. I parametri di *EnableIRQ* sono: un puntatore alla funzione *ISR* che dovrà essere chiamata ogni volta che viene generato un interrupt e un puntatore a *void* che contiene il parametro che verrà passato alla funzione ISR.

La pressione lunga

Moltissimi telecomandi inviano due segnali leggermente differenti se si preme due volte consecutivamente lo stesso tasto, questa tecnica viene usata per distinguere una pressione lunga da due brevi. In molti televisori la pressione lunga viene usata per attivare i canali dal 10 in poi.

Per poter ottenere il puntatore alla funzione ISR è necessario che questa sia un metodo statico. Dal momento che un metodo statico non può accedere al puntatore *this*, non appartenendo ad una istanza specifica della classe, è necessario passare tale puntatore come parametro, in modo che la funzione "conosca" la sua classe. *DisableIRQ*, agisce esattamente all'opposto della funzione *EnableIRQ*, disabilitando il supporto dell'interrupt. È molto importante che la funzione ISR sia il più breve possibile (in termini computazionali) in modo da essere eseguita molto velocemente, contrariamente, infatti, potrebbe verificarsi la sovrapposizione di più di una chiamata a tale funzione, causando evidenti problemi al campionamento e introducendo la necessità di utilizzare metodi di gestione degli accessi ai dati tramite l'uso di oggetti di sincronizzazione come le *sezioni critiche*. Una volta creata la classe *CParallelIO* siamo in grado gestire completamente la porta parallela, bisogna notare che per compilare il codice è necessario anche linkare al progetto la libreria *upp_api.lib*. Per avviare poi il programma bisogna copiare la DLL *upp_api.dll* nella stessa directory dell'eseguibile. Possiamo adesso creare uno strato ulteriore che si occupa del campionamento dei segnali infrarossi attraverso l'uso di un'istanza di *CParallelIO*; a tale scopo creiamo la classe *CIRreceiver*. *CIRreceiver* è sostanzialmente una classe contenente un buffer di interi usato per memorizzare i dati del campionamento, che si serve di *CParallelIO* per gestire le operazioni di input/output attraverso la porta parallela. Il metodo responsabile del campionamento è *StartSampling*. Chiamando questa funzione viene immediatamente cancellato il campionamento precedente tramite la funzione *ClearSamples*, viene attivata l'alimentazione del circuito accendendo il *bit0 (Pin 2)* di uscita e quindi viene attivato l'interrupt tramite il metodo *EnableIRQ* di *CParallelIO*. Analogamente *StopSampling* interrompe il campionamento. Questa classe mette poi a disposizione le funzioni per accedere al buffer dei campioni e due funzioni utili per l'analisi del campionamento.

La prima di queste due funzioni è *GetBreakTime*, una volta ottenuto un campionamento, questa funzione restituisce il valore del *"break time"* ovvero del valore più grande contenuto tra i campioni; questo valore di norma rappresenta la "pausa" tra un segnale ed un altro, tuttavia non sempre questo avviene, dal momento che se il tasto del telecomando viene premuto per un tempo troppo breve viene mandato un solo segnale senza *"break time"*.

La seconda funzione è *CalcSignalLength*, grazie alla quale è possibile ottenere la lunghezza del segnale anche se ne è stato campionato più di uno o è stato interrotto un campionamento proprio durante un segnale. A tale scopo questa funzione esamina il campionamento trovando prima di tutto il *"break time"* e contando quante volte si presenta tra i campioni, taglia i segnali non completi alla fine del buffer e misura la lunghezza apparente del segnale come numero di campioni prima che si verifichi il break time. Infine verifica se, tolti i "break time", il numero di campioni rimanenti è divisibile per la lunghezza apparente del segnale. Questo metodo restituisce quindi la lunghezza del segnale cercando di individuare le situazioni ambigue nelle quali per scarsa pulizia del segnale o per problemi di campionamento non è possibile ricavare questo dato. Ottenere la lunghezza del segnale è importante nel caso in cui si voglia implementare il riconoscimento dei tasti, in questo caso, infatti, bisogna usare un buffer circolare lungo esattamente quanto il segnale e confrontare ad ogni nuovo campionamento con l'archivio di tasti di riferimento che precedentemente si è registrato. Questo metodo è praticamente identico a quello usato nei videogames per il riconoscimento dei cosiddetti cheats, ovvero quelle parole chiave che sbloccano i famosi "trucchi". È bene ricordare sempre che i confronti, così come fa la funzione *CalcSignalLength*, devono essere sempre eseguiti con una certa tolleranza di 1-2 millisecondi per evitare un'eccessiva rigidità. Nel codice sorgente potete trovare la macro EQ_TOLL che verifica l'uguaglianza di due valori considerata una data tolleranza. Per la misurazione degli intervalli di tempo è stato usato quello che viene definito *performance counter*, si tratta di un timer ad altissima frequenza e di conseguenza di ottima precisione. La frequenza di questo timer varia a seconda della velocità del processore e la si può ottenere tramite la funzione *QueryPerformanceFrequency*. Generalmente si hanno frequenze dell'ordine di 3 milioni di cicli per secondo, per questo motivo la misurazione dei tempi in millisecondi è da ritenere piuttosto accurata.

CONCLUSIONI

Abbiamo, quindi, sviluppato un sistema completo, dall'hardware al software, per il riconoscimento di segnali dai telecomandi IR, tutto questo comincia ad avere un senso concreto se ad ogni tasto associamo poi una funzione, ancora meglio se questa funzione può interagire con altri programmi o parametri di sistema. Ad esempio, se un nostro programma, ricevuto il segnale del tasto *Volume*, aumenta davvero il volume di sistema ecc. Interagire con altre applicazioni non è esattamente un lavoro semplice in Windows, bisogna infatti conoscere tutti i parametri della finestra con la quale si vuole interagire, a tale scopo, per eseguire i test, può essere usato il tool Spy++ di Visual Studio. Infine si potrebbe anche pensare di "ritrasmettere", con un'altra periferica, il segnale campionato, per "comandare" il televisore o il videoregistratore dal nostro PC, in questo caso la situazione circuitale si complica leggermente e bisogna avere chiari alcuni concetti non immediati di elettronica e trasmissione di segnali. Spero di avere stimolato la vostra fantasia, non mi resta che salutarvi e augurarvi buona programmazione.

Amedeo Margarese

L'uso dei buffer

Quando si lavora con le periferiche spesso il tempo e la velocità computazionale del software di interfacciamento ricoprono ruoli molto importanti, soprattutto nel caso in cui non vengano usati dei buffer per colmare eventuali differenze di velocità tra il software e l'hardware.

Sistema

☑ Flash Mx e le applicazioni data-oriented.

Firefly: i dati prendono il volo

Libera la tua fantasia, con un kit che collega le applicazioni Flash a qualsiasi fonte di dati.

File sul CD
\soft\codice\firefly.zip

File sul Web
www.ioprogrammo.net /files/72/firefly.zip

Deltapacket
Sono i pacchetti di dati creati dall'oggetto resolver in un formato xml-based che contiene tutte le informazioni necessarie per l'aggiornamento della fonte dati.

È ormai fuori da ogni dubbio che Flash Mx si è rivelato col tempo uno strumento ideale per creare applicazioni dinamiche orientate ai dati. Utilizzare uno strumento come questo significa, per lo sviluppatore, poter creare materiale per il web senza dover stare a preoccuparsi dei problemi inerenti ad incompatibilità tra differenti versioni di browser o di piattaforma, e di poter quindi garantire una diffusione che copre quasi l'intero target del web attraverso il plugin flash player, ormai installato sul 98% delle macchine collegate ad Internet. Inoltre è finalmente possibile oltrepassare gli stretti vincoli che si incontrano nella creazione dell'interfaccia utente dell'applicazione, sviluppando con linguaggi che non permettono di dare libero sfogo alla creatività. Abbiamo visto in più occasioni, su questa rivista, come far interagire Flash con fonti dati esterne:

- l'oggetto **XML**: è possibile integrare i dati in Flash con i server che usano la tecnologia XML per creare applicazioni sofisticate utilizzando i metodi dell'oggetto.

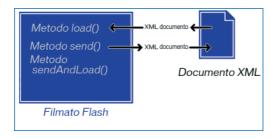



Fig. 1: La comunicazione con l'oggetto XML di ActionScript.

- l'oggetto **LoadVars:** è possibile trasferire le variabili da un filmato Flash a un server e viceversa attraverso l'utilizzo dell'oggetto *LoadVars*, che consente di inviare tutte le variabili in un oggetto a un URL specificato e caricare tutte le variabili di un URL specificato in un oggetto.
- **Flash Remoting**: mette a disposizione l'infrastruttura per connettersi a servizi remoti messi a disposizione dagli application server. Questo significa poter invocare metodi da un servizio creato con Coldfusion Mx, J2EE o .NET.

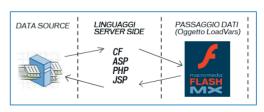



Fig. 2: La comunicazione con l'oggetto loadVars di ActionScript.

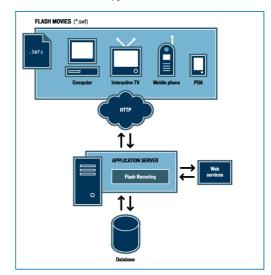



Fig. 3: Funzionamento della tecnologia del Flash Mx Remoting.

Con questi metodi possiamo praticamente accedere a qualunque fonte dati esterna, intercettare i dati dinamicamente, inserirli nell'interfaccia grafica per l'utente e passare i dati processati di nuovo all'oggetto remoto.

FLASH MX DATA CONNECTION KIT: FIREFLY COMPONENTS

Nonostante in Flash Mx fosse già semplice l'accesso a oggetti remoti, con i Firefly components il tutto si riduce al trascinamento di oggetti già funzionanti sullo *Stage* e nella compilazione dello loro proprietà attra-

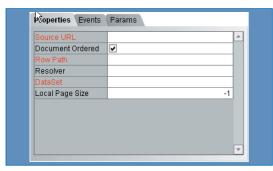



Fig. 4: **La finestra per configurare i parametri dei Firefly Components.**

verso la finestra *"Parametri dei Componenti"*. I vantaggi nell'utilizzo di questa suite sono molteplici:

- *Non richiedono software aggiuntivo*: sviluppare con questa suite di components significa muoversi all'interno dell'ambiente di Flash Mx e scrivere codice Actionscript. La curva di apprendimento di questa nuova tecnologia è molto rapida, anche grazie alla ricca documentazione installata insieme al pacchetto.
- *Compatibili con il flash player*: dal lato utente, navigare all'interno di un'applicazione creata con i *Firefly Components* non comporta lo scaricamento di player proprietari, in quanto il file pubblicato dallo sviluppatore consiste in un *swf* fruibile dal browser attraverso il classico plugin del flash player.
- *Ambiente di sviluppo visuale*: la configurazione dei componenti avviene tramite una finestra di settaggio dei parametri dell'oggetto utilizzato. In questo modo tutte le proprietà sono modificabili senza inserire codice ActionScript.
- *Indipendenza dalla fonte dati*: lo sviluppatore potrà creare la sua applicazione senza preoccuparsi di conoscere la tecnologia utilizzata per l'accesso alla fonte dati. XML, SQL Server 2000 o Flash Remoting sono tutti data sources facilmente collegabili agli oggetti che compongono la nostra interfaccia utente attraverso i *Firefly components*.
- *Separazioni dei contenuti dall'interfaccia utente*: essere indipendenti dalla fonte dati significa poter finalmente svincolare i contenuti dinamici della nostra applicazione dall'interfaccia utente.

I Firefly mettono a disposizione degli sviluppatori una architettura flessibile per accedere, editare e salvare dati provenienti da differenti data sources. L'engine lavora con i dati lato server, utilizzando tre componenti specifici per compiere le operazioni relative al recupero dei dati, alla visualizzazione e al salvataggio. Questi oggetti sono denominati *Data Access Components* e sono rispettivamente:

- **Connector**: è responsabile per il recupero dei dati dalla fonte dati remota. A seconda della tecnologia utilizzata, lo sviluppatore avrà a disposizione il relativo oggetto *Connector*. I dati che riceve vengono caricati lato client e passati al Dataset.
- **Dataset**: si occupa di mappare la complessa struttura dati in una più semplice configurazione formata da una struttura del tipo righe/colonne. In questo modo, i dati possono essere facilmente manipolati, indicizzati, filtrati e formattati nell'interfaccia utente.
- **Resolver**: è responsabile per il salvataggio dei dati inseriti o modificati dall'utente. Anche per quest'oggetto l'utente ne ha a disposizione differenti a seconda della fonte dati utilizzata. Una volta che i dati vengono salvati, il *Resolver* li converte in un formato proprietario basato su XML e denominato *Deltapackets*.

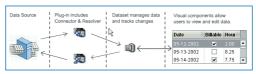



Fig. 5: **L'architettura dei Firefly.**

Affinché i dati vengano esposti all'utente formattati all'interno dell'interfaccia dell'applicazione, i Firefly mettono a disposizione una suite di componenti chiamata *DataLink Components* e composta da: *FxGrid*, *FxCheckBox*, *FxComboBox*, *FxEditBox*, *FxListBox*, *FxLookup ComboBox*, *FxLookup ListBox*, *FxMemoBox*, *FxNavigator.* Questi componenti permettono allo sviluppatore di avere a disposizione elementi di interfaccia pronti per l'uso e configurabili, tramite appositi pannelli di proprietà, capaci di formattare i dati passati dall'oggetto *Dataset.*

UTILIZZARE I FIREFLY

Dopo aver installato il pacchetto dei Firefly Components, lo sviluppatore è in grado di creare le sue applicazioni. Aprendo Flash Mx notiamo immediamente che la palette di destra, sotto la voce *Componenti*, è stata arricchita di quattro voci : *Firefly Components*, *Firefly Remoting*, *Firefly Sql*, *Firefly XML.* Per poter meglio capire dove veramente risiede la potenza e la flessibilità del pacchetto, andremo a creare un'applicazione web che caricherà dinamicamente i dati relativi al catalogo corsi di una società fittizia. Una griglia mostrerà le informazioni relative ai corsi disponibili, e dei campi testo mostreranno i prezzi e le tariffe orarie del corso selezionato dall'utente. Un sistema di navigazione permetterà al cliente di spostarsi all'interno dei dati caricati. La Fig. 7 illustra meglio di ogni altra spiegazione come è formata la nostra interfaccia utente. A scopo puramente didattico, proveremo la nostra applicazione caricando prima i dati da un file XML e successivamente mantenendo inalterata la *UI (User Interface)*, e senza scrivere una riga di codice aggiuntivo, usufruiremo del Flash Remoting per connetterci ad un database in Access. La finalità dell'esercizio è quella di far notare come, cambiando semplicemente

Sistema

Firefly:
i dati prendono il volo

Ordine dei layer

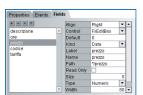
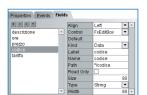
È importante fare attenzione nell'inserimento dei vari tipi di componenti nei layer. In particolare gli oggetti Connector, Dataset e Resolver vanno inseriti in un layer sottostante a quello dei componenti per non generare errori. Questo perché non possono essere visualizzati i dati se prima non vengono caricati.

Componenti lato authoring

I componenti Connector, Resolver e Dataset sono componenti visibili solo in lato authoring. Quando infatti andiamo ad eseguire il filmato l'utente non li visualizzerà sullo Stage.

Sistema

Proprietà del campo "descrizione".

Proprietà del campo "ore".

Proprietà del campo "prezzo".

Proprietà del campo "codice".

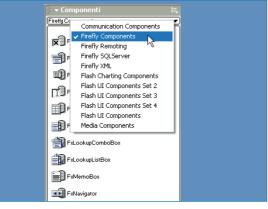



Fig. 6: Il pacchetto Firefly, una volta installato, arricchisce di 4 voci il pannello componenti.

il componente *"Connector"* non dovremo modificare nulla nell'interfaccia grafica né tantomeno preoccuparci di dover riformattare i dati.

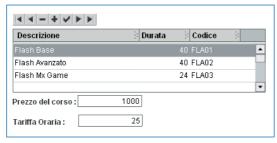



Fig. 7: L'interfaccia utente della nostra applicazione.

CREIAMO LA CONNESSIONE ALLA FONTE DATI

Nel paragrafo precedente abbiamo detto che la fonte dati remota da cui caricare i dati risiede in un file XML (salvato col nome di *corsi.xml*) così composto:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<formazione>
        <corso id="1">
                <codice>FLA01</codice>
                <ore>40</ore>
                <prezzo>1000</prezzo>
                <descrizione>Flash Base</descrizione>
        </corso>
```

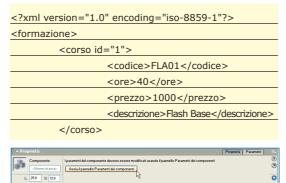



Fig. 8: Il bottone per aprire il pannello dei parametri dei componenti.

Il codice completo del file XML lo trovate nel file *corso.xml* nel CD-Rom allegato alla rivista. Apriamo *Flash Mx* e dalla palette *Componenti* apriamo il menù a tendina e selezioniamo la voce *"Firefly XML"*. Trasciniamo sullo stage l'oggetto *FxXMLConnector* e dal pannello *Proprietà* clicchiamo sulla voce *"Avvia il pannello dei Parametri dei componenti"*. In questa finestra andiamo a configurare il primo tab (*Properties*) dell'ogeto compilando i campi di testo con le seguenti informazioni:

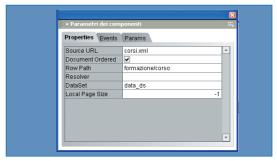



Fig. 9: Il pannello di proprietà dell'oggetto Connector per una fonte dati XML.

- **Source URL**: contiene l'indirizzo relativo o assoluto di un file XML o di un oggetto a cui accedere e che può ritornare informazioni in formato XML. Nel nostro esempio inseriamo il path relativo del file XML *"corsi.xml"*. Il file deve risiedere nella stessa cartella di pubblicazione del *swf*.
- **Document Ordered**: questa proprietà ritorna un valore booleano (*TRUE* o *FALSE*) ed indica se il nostro file XML contiene record ordinati. Avere dati ordinati o indicizzati permette di velocizzare l'accesso ai dati remoti. Clicchiamo sulla check e rendiamola selezionata.
- **Row Path**: identifica il nodo di partenza da cui verranno caricati i dati. Si tratta di una sintassi in *Xpath*. Nel nostro esempio, per farci leggere i record contenuti nel nodo *<corso>*, insieriamo la stringa: *formazione/corso.*
- **Resolver**: contiene il nome di istanza dell'oggetto *Resolver* se presente. Nel nostro caso non è stato inserito tale oggetto. In questo caso i nostri dati sono solo *read-only.*
- **Dataset**: contiene il nome di istanza dell'oggetto *Dataset*. Inseriamo il nome *"data_ds"* del Dataset che tra breve andremo ad inserire.

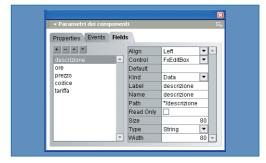



Fig. 10: Il pannello di proprietà dell'oggetto Dataset.

- **Local Page Size**: stabilisce il numero dei record che vengono mappati nel dataset ogni volta che una richiesta di ulteriori record viene eseguita. Lasciamo il valore di default a *–1* In questo modo risultano mappati tutti i dati nel momento in cui viene caricato il file xml.

Questo componente ha il compito di collegarsi alla fonte dati, ritirare i record dal file xml specificato nel parametro *Source url* e restituire il tutto al Dataset. Inseriamo quindi questo oggetto sullo Stage trascinando la voce *FxDataset* dalla palette *Componenti*, sotto la voce *Firefly Components*. Dal pannello in basso *Proprietà* settiamo il nome di istanza a: *data_ds* e avviamo il pannello dei *Parametri del componente*. Clicchiamo sul terzo tab, *Fields*, e creiamo le associazioni con i dati ottenuti dal *Connector* con il tastino +. Inseriamo cinque nuovi field, che rappresentano i dati da inserire nella nostra interfaccia, e settiamo le loro proprietà. Ogni field ha le seguenti proprietà da settare:

- **Align** - allineamento del testo.
- **Control** - a quale componente datalink assoceremo il dato.
- **Default** - il valore di default del dato.
- **Kind** - il tipo di dato ricevuto.Possiamo scegliere tra *Data*, *Calculated*, e *Lookup*.
- **Label** - è il nome che utilizzeremo per collegare il dato dinamico al componente.
- **Name** – il nome del componente che ci permette di accedere ad esso tramite Actionscript.
- **Path** - indica con una sintassi in Xpath il nodo da cui leggere il dato.
- **ReadOnly** - indica se il dato può essere modificato o se è di sola lettura.
- **Size** - proprietà applicabile ai dati di tipo stringa per determinare la loro lunghezza.
- **Type** - il formato del tipo didato ricevuto (es., String, Money, Date, etc.).
- **Width** - rappresenta la larghezza del componente che contiene il dato.

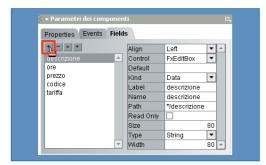



Fig. 11: Mappiamo i dati ricevuti dal Connector dal tab Fields.

I fields che creeremo sono: *descrizione, ore, prezzo, codice, tariffa* (è un dato di tipo *calculated*, cioè creato in runtime via codice) e settiamo le loro proprietà così come indicato nelle figure a lato. La parte relativa al caricamento dei dati in remoto è ultimata, ora rimane da creare l'interfaccia utente dell'applicazione.

L'INTERFACCIA UTENTE: I DATALINK COMPONENTS

Dalla palette *Componenti* apriamo il menu a tendina e selezioniamo la voce *"Firefly Components"*. Trasciniamo sullo Stage i seguenti oggetti disponendoli come in Fig. 7: *FxNavigator, FxGrid, 2 FxEditBox*. Selezionado uno alla volta i componenti accediamo e compiliamo le loro proprietà dal pannello *Proprietà* in basso come segue:

- **FxNavigator**: settiamo la proprietà *"Dataset Name" = data_ds* (nome di istanza del nostro oggetto Dataset).
- **FxEditBox** *(relativo al prezzo)*: settiamo la proprietà *"Dataset Name" = data_ds* e la proprietà *FieldName = prezzo* (è la label che gli abbiamo associato nel tab *"fields"* del Dataset).
- **FxEditBox** *(relativo alla tariffa oraria)*: settiamo la proprietà *"Dataset Name" = data_ds* e la proprietà *FieldName = tariffa.*
- **FxGrid** *(relativo al prezzo)*: dal tab *"Properties"* settiamo la proprietà *"Dataset Name" = data_ds*. Clicchiamo sul tab *"Columns"* e inseriamo tre nuove colonne come nelle relative figure.

A questo punto salviamo il filmato, assicuriamoci che il file *corsi.xml* sia presente nella cartella, copiamo dentro i file *.swf* che trovate nella cartella *"[drive]:\Programmi\Macromedia\Flash MX\Configuration\Firefly\Lib"* ed eseguiamo il filmato. L'esempio completo, che si trova in allegato al CD-Rom della rivista, comprende anche il codice necessario per la visualizzazione del campo (tipo *Calculated*) *"Tariffa Oraria"* che consiste in una divisione tra il prezzo del corso e la durata:

```
getFieldByName("tariffa").setAsNumber(
  getFieldByName("prezzo").getAsNumber() / getFieldBy-
                                Name("ore").getAsNumber() );
```

Se a questo punto volessimo cambiare la fonte dati ed utilizzare il Flash Remoting per accedere ad un database Access che contiene le stesse informazioni, basta cambiare l'oggetto *FxXMLConnector* con l'oggetto *FxRecordsetConnector* e settare le sue relative proprietà come nell'esempio allegato (*remoting_2_OK.fla*).

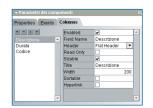
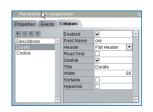
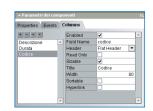
AD OGNUNO IL SUO LAVORO

Ora più che mai il confine tra i compiti inerenti alla figura del Web designer e del Web developer risulta marcato. Creare applicazioni orientate ai dati con Flash Mx diventa ancora più semplice con il *Data Connection Kit*. Lo sviluppatore non dovrà più interessarsi di sapere a priori da dove provengono i dati o avere il terrore che durante la lavorazioni il datasource migri verso un'altra tecnologia. Abbiamo finalmente la certezza che l'interfaccia grafica della nostra applicazione non subirà più modifiche e sarà completamente svincolata dai contenuti. Flash Remoting, XML, SQL Server 2000 mettono la loro potenza di *data management* a disposizione della semplicità d'uso dei *Firefly components* per diventare tecnologie alla portata di tutti.

Marco Casario

Sistema

Firefly:
i dati prendono il volo

Proprietà del campo "tariffa".

Proprietà della colonna "descrizione".

Proprietà della colonna "durata".

Proprietà della colonna "codice".

Deployment ed upgrade automatico di applicazioni .NET

Sistema

Applicazioni auto-aggiornanti

In questo articolo viene esposta una breve panoramica dei miglioramenti più significativi introdotti dal Microsoft .NET Framework per la fase di deployment e mostrato come sia possibile realizzare applicazioni .NET che effettuano l'upgrade "on the fly".

File sul CD
\soft\codice\AutoUpgrade.zip

File sul Web
www.ioprogrammo.net/files/72/AutoUpgrade.zip

Windows 2000
A partire da Windows 2000, si è cercato di introdurre nuove soluzioni, come funzionalità Syde-by-Syde, al fine di attribuire ad ogni applicazione una propria versione di DLL. Tuttavia, prima dell'introduzione di .NET, Microsoft non ha proposto soluzioni significative per la risoluzione del problema.

Lo sviluppo di applicazioni per Windows ha sempre comportato una serie di problemi che esulano dalla specificità della programmazione pura. Sia programmatori esperti, sia utenti finali, sono a conoscenza della possibilità di trovarsi dinanzi ad applicazioni che cessano di funzionare dopo l'installazione di un nuovo programma. Microsoft stessa è arrivata a definire il fenomeno *DLL Hell* ovvero *"L'inferno delle DLL"*. Il problema nasceva a causa della mancanza di un meccanismo che tenesse traccia delle versioni di DLL. Spesso poteva verificarsi la spiacevole situazione in cui l'installazione di un programma sovrascriveva una determinata DLL, condivisa da più programmi, con una versione differente. I problemi sorgono sia nel caso in cui la libreria dinamica appena installata sia obsoleta per alcune applicazioni condivise, sia quando la libreria è troppo recente e non garantisce compatibilità verso il basso. A partire da Windows 2000, si è cercato di introdurre nuove soluzioni, come funzionalità *Syde-by-Syde*, al fine di attribuire ad ogni applicazione una propria versione di DLL. Tuttavia, prima dell'introduzione di .NET, Microsoft non ha proposto soluzioni significative per la risoluzione del problema e, ad onor del vero, sono stati introdotti ulteriori problemi con le *DLL COM*. Purtroppo, ogni possibile soluzione non è sufficiente a colmare la lacuna di un'architettura priva di verifica delle versioni e dipendenze.

LA RISPOSTA .NET

Microsoft .NET Framework introduce una nuova architettura per lo sviluppo di *Windows Forms*, ovvero di applicazioni Windows aventi interfacce user friendly, integrate e interagenti col sistema operativo sottostante. Microsoft, che ha investito moltissimo su questa nuova tecnologia, cerca di ridurre al minimo il *TCO (Total Cost of Ownership)* delle aziende che sviluppano con .NET. La fasi di deployment e manutenzione sono da sempre un ostacolo per le applicazioni Windows, e hanno favorito il diffondersi di applicazioni web-based, soprattutto in ambito distribuito. Microsoft .NET è una risposta efficace a questi problemi. Le applicazioni .NET sono costituite da blocchi logici e fisici, detti *assembly*. Ogni assembly è un'unità di deployment per i tipi e le risorse, può essere composto da uno o più file, e può essere sia eseguibile *(.exe)* sia una libreria *(.dll)*. Gli *assembly* possono essere *privati o condivisi* a seconda del numero di applicazioni che ne dipendono. Si parla di *assembly privato* se un file eseguibile *.exe* necessita di una libreria di classi *.dll* che non è necessaria per il funzionamento di altre applicazioni. L'installazione di applicazioni contenenti assembly privati, richiede una semplice operazione di *xcopy* dei file in una directory qualsiasi della macchina client. Questo tipo di installazione che va sotto il nome di *"installazione a impatto zero"* o *Xcopy Deployment* non richiede la registrazione di alcuna chiave nel registro di Windows. Il meccanismo di Xcopy Deployment non prevede purtroppo la possibilità di creare icone sul desktop, nel menu *Start* di Windows o di aggiungere una entry nel panello *Installazione Applicazioni* (*Aggiungi/Rimuovi programmi*), così come di inserire la possibilità di riparare l'installazione,

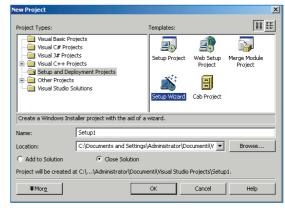



Fig. 1: La finestra di dialogo per la creazione di un progetto di setup in Visual Studio .NET 2003.

come avviene per i software commerciali più diffusi. Con questo tipo di esigenze è indispensabile l'uso di software per la realizzazione di installazioni, o la creazione di un progetto di deployment in Visual Studio .NET. Nel caso di assembly condivisi da più applicazioni, bisogna avere maggiore cura nel definire in maniera esatta il nome e il numero di versione, è dunque necessario seguire una serie di regole e convenzioni e solitamente vengono installati nella *GAC (Global Assembly Cache)*. Gli assembly rappresentano una brillante soluzione al DLL Hell, principalmente grazie alla loro caratteristica di essere *auto-descrittivi* e di mantenere, all'interno del *manifesto* dell'assembly, le dipendenze dalla versione. Le applicazioni .NET, inoltre, vengono isolate attraverso il meccanismo dei *domini delle applicazioni* garantendo l'esecuzione di più applicazioni all'interno di un unico processo suddiviso in più domini. Effettuare l'upgrade di un'applicazione, può voler dire, nella maggior parte dei casi, la sostituzione di alcuni file con delle versioni più aggiornate. Un'applicazione che effettua il caricamento di alcuni assembly, li pone in uno stato di lock, che impedisce l'aggiornamento o la copia di nuove versioni tramite xcopy. Una prima soluzione a questo problema è data da un meccanismo detto *Shadow Copying*. Un piccolo eseguibile, che rimane in lock durante l'esecuzione, si occupa di creare un oggetto dominio di applicazione per il programma principale ed assegnare alla proprietà *ShadowCopyFiles* dell'oggetto *AppDomainSetup* il valore string *"true"*. In questo modo .NET crea una copia *"shadow"* dei file di installazione e permette operazioni di ricompilazione on the fly, o aggiornamento dei nuovi assembly che saranno caricati solo al riavvio dell'applicazione stessa. Seguono le poche righe di codice necessarie per l'eseguibile che si incarica di avviare il programma principale:

```
<CODICE VB.NET>
Imports System
Module ShadowInstall
   Class Avvia
      Sub Main()
         'crea un oggetto AppDomainSetup
         Dim ds = New AppDomainSetup()
         'attiva la modalità Shadow Copy
         ds.ShadowCopyFiles = "true"
           'crea il dominio ed esegue l'assembly
                                        Programma.exe
          AppDomain.CreateDomain("EsempioShadow",
                   AppDomain.CurrentDomain.Evidence,ds
    ).ExecuteAssembly("nomedirectory\Programma.exe")
      End Sub
   End Class
End Module
</CODICE VB.NET>
```

```
<CODICE C#>
using System;
namespace ShadowInstall
{
   public class Avvia
  {
   public static void Main()
   {
      //crea un oggetto AppDomainSetup
      AppDomainSetup ds = new AppDomainSetup();
      //attiva la modalità Shadow Copy
      ds.ShadowCopyFiles = "true";
      //crea il dominio ed esegue l'assembly Programma.exe
       AppDomain.CreateDomain("EsempioShadow",
                  AppDomain.CurrentDomain.Evidence,ds)
    .ExecuteAssembly("nomedirectory\\Programma.exe");
    }
  }
}
</CODICE C#>
```

```
<CODICE J#>
package ShadowInstall;
import System.*;
public class Avvia
{
  public static void main()
  {
     //crea un oggetto AppDomainSetup
     AppDomainSetup ds = new AppDomainSetup();
     //attiva la modalità Shadow Copy
     ds.set_ShadowCopyFiles("true");
     //crea il dominio ed esegue l'assembly Programma.exe
      AppDomain.CreateDomain("EsempioShadow",
      AppDomain.get_CurrentDomain().get_Evidence(),ds)
    .ExecuteAssembly("nomedirectory\\Programma.exe");

  }
}
</CODICE J#>
```

WEB BASED DEPLOYMENT

In molte applicazioni reali viene a determinarsi l'esigenza di fornire ai propri utenti una versione più aggiornata del programma sviluppato. Una delle possibili soluzioni è il deployment basato sul Web. Questo ti-

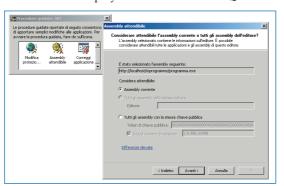



Fig. 2: **Le security policy del .NET Framework per l'attendibilità di un'applicazione distribuita via browser.**

Sistema

Applicazioni
auto-aggiornanti

Sul Web

Seguono una serie di interessanti articoli e link di approfondimento.

Simplifying Deployment and Solving DLL Hell with the .NET Framework
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/dplywithnet.asp

Deploying .NET Applications: Lifecycle Guide
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/DALGIssues.asp

Death of the Browser?
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnadvnet/html/vbnet10142001.asp

Deploying a Runtime Application Using Internet Explorer
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpcondeployingcommonlanguageruntimeapplicationusingie55.asp

.NET Framework Deployment Guide
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetdep/html/dotnetframedepguid.asp

Deploying application
http://msdn.microsoft.com/library/default.asp?url=/library/enus/cpguide/html/cpcondeployingnetframeworkapplications.asp

Sistema

Applicazioni
auto-aggiornanti

po di deployment prevede due possibilità, solamente la prima richiede l'ausilio del browser dell'utente. L'applicazione può essere avviata mediante Internet Explorer inserendo il corretto URL dell'eseguibile principale. .NET provvede questo tipo di deployment che per funzionare richiede la configurazione delle policy di sicurezza in modo che il codice dell'eseguibile e gli altri assembly correlati vengano considerati attendibili dal framework. Entrambe le soluzioni di distribuzione presentano pesanti limitazioni, ma in particolar modo può risultare fastidioso dover avviare un'applicazione tramite browser. Il secondo metodo web-based consiste nel distribuire all'utente un eseguibile dalle dimensioni ridotte ed in grado di caricare gli assembly da un web server specificato. Questo secondo metodo evita la scomodità di dover avviare il browser per eseguire il programma e ha un'implementazione piuttosto immediata. L'eseguibile crea e carica l'istanza della classe di "avvio" del programma attraverso il metodo *LoadFrom* della classe *Assembly*. Gli altri assembly necessari per le altre classi del programma saranno gestiti in maniera automatica dal framework. In questo modo l'applicazione caricherà gli assembly sempre nella loro versione più aggiornata. Segue lo script necessario all'eseguibile distribuito ai client:

```
<CODICE VB.NET>
Imports asm = System.Reflection.Assembly
...
Dim location As String = "http://azienda/applicazione1
        /mioprogramma.dll" CType(Activator.CreateInstance(
    asm.LoadFrom(location).GetType("ShowInstall.Avvia")
                        ), Windows.Forms.Form).Show()
...
</CODICE VB.NET>

<CODICE C#>
using asm = System.Reflection.Assembly;
...
string location = "http://azienda/applicazione1
                                /mioprogramma.dll";
  ((Form) Activator.CreateInstance(asm.LoadFrom(location)
.GetType("ShowInstall.Avvia"))).Show();
...
</CODICE C#>
```

```
<CODICE J#>
import System.Reflection.*;
...
String location = "http://azienda/applicazione1
                                /miopogramma.dll";
  ((Form) Activator.CreateInstance(
                        Assembly.LoadFrom(location)
.GetType("Forms.Startup"))).Show();
...
</CODICE J#>
```

I limiti di queste due tecniche sono legati alla necessità di essere collegati ad Internet. I programmi non potranno essere avviati se non si è connessi al web server, e questo può essere un grande ostacolo nell'implementazione di soluzioni reali; inoltre i permessi concessi alle applicazioni sono piuttosto ridotti rispetto alle applicazioni installate stabilmente nel sistema operativo ed il caricamento automatico degli aggiornamenti non è esteso a file che non siano assembly o file di configurazione. Le soluzioni di deployment web-based sono adeguate nei casi in cui l'applicativo è presente in una Lan connessa a internet o al proprio web server, e l'aggiornamento è necessario solo per file .NET, escludendo quindi script, help, file xml, ecc…ecc…

XCopy Deployment

.NET introduce diverse caratteristiche interessanti per la distribuzione delle applicazioni. Una delle più importanti è l'*Xcopy Deployment*.
Xcopy sta per *"Extended Copy"* ed è un comando MS Dos che effettua la copia di tutti i file e delle relative sottodirectory presenti in una directory di base. .NET riprende il termine *"xcopy"* poiché, grazie all'ausilio di un'architettura basata su assembly auto-descrittivi, è possibile installare un programma (in managed code) semplicemente copiando tutti i file contenuti nella directory dell'applicazione in una cartella del computer di destinazione. Questo tipo di distribuzione degli applicativi è anche detta *"Installazione a impatto zero"*.

LA CLASSE AUTOUPGRADE

Il Framework .NET non fornisce una classe o una libreria che agevoli l'upgrade automatico oltre quanto visto sinora. Le potenzialità di .NET, tuttavia, permettono la creazione di soluzioni complesse in maniera ben organizzata, efficiente e rapida. Esistono diversi componenti, framework, librerie sviluppati in .NET, che rendono l'architettura di sviluppo ancora più ricca. Le funzionalità di aggiornamento automatico possono essere abilitate attraverso un'apposita classe *AutoUpgrade* che viene fornita nella libreria di classi *AutoUpgrade.Lib.dll*, sviluppata in VB.NET da Anthony Glenwright. La classe si occupa di confrontare le versione o la data di ultima modifica dei file che costituiscono l'applicazione. Ciò è possibile grazie al confronto tra il manifesto xml dell'applicazione locale e quello reperito dal web server. La classe, inoltre, si occupa della generazione del manifesto e dell'upgrade nel caso in cui vi siano dei file sul server più recenti di quelli dell'applicazione del client. La libreria di classi e la documentazione completa in versione originale (in inglese) sono presenti nel CD allegato alla rivista (liberamente utilizzabili).

AGGIORNAMENTO AUTOMATICO DELLE APPLICAZIONI

L'inserimento di funzionalità di aggiornamento automatico, sfruttando la classe *AutoUpgrade*, è un'opera-

Fig. 3: Il riferimento alla libreria *AutoUpgrade.Lib .dll* nel Solution Explorer di Visual Studio .NET 2003.

zione davvero semplice e implementabile con poche righe di codice. Se si sta sviluppando con Visual Studio .NET è necessario creare un riferimento alla libreria *.dll* nel Solution Explorer (*Esplora Soluzioni*) come mostrato in Fig. 3.
L'applicazione aggiornata andrà posta sul web server in una directory dalla quale saranno prelevati tutti i file dell'applicazione. L'eseguibile principale non si occupa del vero e proprio aggiornamento, ma determina se è necessario un aggiornamento o meno ed eventualmente avvia un piccolo eseguibile *AutoUpgrade.exe* che si occupa di effettuare il download dei file aggiornati.
Seguono alcune righe di codice che possono essere aggiunte alle applicazioni per dotarle di controllo per l'aggiornamento:

```
<CODICE VB.NET>
Dim autoupg As AutoUpgrade()
Dim locationman As String =
          "http://azienda/applicazione1/miomanifesto.xml"
autoupg = AutoUpgrade.Create(locationman)
If Not autoupg.IsUpgradeAvailable(True) Then
     System.Windows.Forms.Application.Run(new Form1)
Else
  'termina l'applicazione
End If
</CODICE VB.NET>
```

```
<CODICE C#>
string locationman = "http://azienda/applicazione1
                                    /miomanifesto.xml";
AutoUpgrade autoupg = AutoUpgrade.Create(
                                          locationman);
if (!autoupg.IsUpgradeAvailable(true)) {
    System.Windows.Forms.Application.Run(new Form1());
else{
   //termina l'applicazione
}
</CODICE C#>
```

```
<CODICE J#>
String locationman = "http://azienda/applicazione1
                                    /miomanifesto.xml";
AutoUpgrade autoupg = AutoUpgrade.Create(locationman);
if (!autoupg.IsUpgradeAvailable(true)) {
   System.Windows.Forms.Application.Run(new Form1());
}
else{
  //termina l'applicazione
}
</CODICE J#>
```

La classe *AutoUpgrade* fornisce vari overload del metodo *Create*. Nel nostro caso *Create(locationman)* ottiene il manifesto aggiornato presente all'indirizzo specificato e crea un'istanza della classe *AutoUpgrade*.
Nel caso in cui il percorso specificato non sia valido o non sia presente un manifesto, l'applicazione continuerà a funzionare correttamente senza alcuna operazione di aggiornamento, poiché l'istruzione *!autopg .IsUpgradeAvailable(true)* risulterà *true*. Infatti, *IsUpgradeAvaible* ha il compito di verificare se il manifesto sul server è più recente di quello in locale; il parametro (*true*) permette l'invocazione del metodo *StartUpgradeStub* per l'avvio del programma di aggiornamento.
Se si desidera gestire in maniera manuale le operazioni di aggiornamento sfruttando i metodi di più "basso livello" presenti nella classe, si può porre a *false* tale parametro. Per la distribuzione sarà sufficiente inserire nella directory del client i file *AutoUpgrade.exe*, *AutoUpgrade.Lib .dll* e gli assembly del programma vero e proprio. L'utente avvia l'applicazione, il codice precedente controlla se è presente un upgrade da effettuare, in caso favorevole salva il nuovo manifesto ed esegue *AutoUpgrade.exe* che effettua il download dei file aggiornati dalla directory presente sul web server. Nel caso in cui invece non sia richiesto alcun tipo di aggiornamento o l'utente non sia connesso ad internet, l'esecuzione del programma continuerà normalmente con gli assembly correntemente installati.
Sfruttando la classe *AutoUpgrade*, nel file *AutoUpgrade.exe* si potranno inserire metodi quali *GenerateManifest* e *Save* per la generazione ed il salvataggio del nuovo manifesto xml e, nel caso in cui lo si desideri, gestire individualmente le modalità di download e i singoli file per la copia dei nuovi assembly.
Per maggiori informazioni circa la classe *AutoUpgrade* si veda la documentazione fornita dal produttore ed allegata nel CD della rivista.

CONCLUSIONI

In questo articolo abbiamo visto le novità introdotte dalla tecnologia Microsoft .NET per quanto concerne il deployment e l'aggiornamento di applicazioni on the fly. I vantaggi introdotti dalla possibilità di fornire aggiornamenti al software preesistente sono innumerevoli, e in un'ottica aziendale permettono una riduzione di costi piuttosto interessante. L'aggiornamento può essere inoltre utilizzato per installazioni multiple da un server a più client, risolvendo il caos delle versioni eterogenee che spesso si creano a causa di ben noti problemi di cui abbiamo parlato all'inizio dell'articolo.
In questo articolo abbiamo visto le novità introdotte dalla tecnologia Microsoft .NET per quanto concerne il deployment e l'aggiornamento di applicazioni on the fly. I vantaggi introdotti dalla possibilità di fornire aggiornamenti al software preesistente sono innumerevoli, e in un'ottica aziendale permettono una riduzione di costi piuttosto interessante.
L'aggiornamento può essere inoltre utilizzato per installazioni multiple da un server a più client, risolvendo il caos delle versioni eterogenee che spesso si creano a causa di ben noti problemi di cui abbiamo parlato all'inizio dell'articolo.

Antonio Cangiano

Sistema

Bibliografia

• MICROSOFT .NET PROGRAMMAZIONE AVANZATA
Jeffrey Richter

Pagine: 640
Prezzo: € 60.00
ISBN 88-8331-368-2
30 aprile 2002

Questo testo è l'ideale per chiunque conosca i concetti di programmazione object-oriented (OOP) come l'astrazione dei dati, l'ereditarietà, il polimorfismo. Il libro spiega dettagliatamente il sistema estensibile dei tipi di dato di .NET Framework, esamina come il runtime amministra il comportamento dei tipi di dato, e analizza come un'applicazione li gestisce. Parlando di C#, presenta i concetti applicabili a tutti i linguaggi di programmazione che hanno come target .NET Framework.
Tra gli argomenti trattati:

- **L'architettura di .NET Framework**
- **La costruzione, la distribuzione e l'amministrazione delle applicazioni e i loro tipi di dato**
- **La costruzione e la distribuzione delle risorse condivise**
- **I tipi di dato fondamentali**
- **Lavorare con il testo**
- **Le interfacce e gli attributi**

☑ Ottimizzazione dell'accesso ai dati.

Stored Procedure con ADO.NET e SQL Server

.NET e DB

Realizzare opportune stored procedure può migliorare sensibilmente le prestazioni di applicazioni che accedono ai dati. Questo mese vedremo come realizzare stored procedure in Visual Basic usando ADO .NET e SQL Server.

In questo articolo ci occuperemo di un aspetto particolarmente importante per la progettazione di database professionali: le stored procedure. In particolare vedremo:

- l'utilità delle stored procedure
- come creare stored procedure
- diversi tipi di stored procedure
- funzioni
- come invocare stored procedure con VB/ADO .NET

PERCHÉ STORED PROCEDURE

Per capire bene l'utilità e i motivi che hanno portato all'introduzione delle stored procedure, sarà necessario analizzare il processo di esecuzione di una query all'interno del server di database. Nel momento in cui il DBMS riceve una query SQL, saranno attivate le seguenti operazioni:

- Lettura del codice SQL da parte di un parser che ne verifica la correttezza sintattica.
- Controllo di contesto (context check), cioè verifica dell'esistenza delle entità referenziate nella query (tabelle, viste, attributi, ...).
- Controllo dei diritti d'accesso dell'utente che ha inviato la richiesta.
- Ottimizzazione della query. Un componente del DBMS, il "query optimizer", traduce la query SQL in un preciso algoritmo ed esegue le ottimizzazioni del caso.
- Compilazione dell'algoritmo generato dal query optimizer in linguaggio macchina.
- Esecuzione della query compilata.

In molti casi, il 90% del tempo d'elaborazione si perde nelle fasi che precedono l'esecuzione vera e propria della query. Definendo la query all'interno di una stored procedure, i passi precedenti l'esecuzione sono effettuati solo quando la procedura è memorizzata (o aggiornata) per la prima volta. Le successive invocazioni saranno effettuate a partire dal codice precompilato nel linguaggio macchina nativo. Questo migliora sensibilmente le prestazione del sistema. Non esiste un linguaggio standard per scrivere stored procedure. Ogni server di database ha il proprio ed in generale esso è una combinazione di istruzioni SQL con altri costrutti classici come *IF..THEN..ELSE, FOR, WHILE,* eccetera. E' possibile dichiarare e avvalersi dell'uso di variabili con relativi tipi di dato. Molti DBMS consentono anche di scrivere stored procedure mediante altri linguaggi di programmazione tipo Java SQL Embedded o C++ SQL Embedded. In generale un linguaggio SQL Embedded presenta la stessa sintassi del linguaggio originale (Java, C++, Cobol) ma con estensioni mirate ad "incastrare" istruzioni SQL all'interno del codice. Un preprocessore dedicato convertirà un'applicazione SQL Embedded in un'applicazione compatibile con il compilatore del linguaggio originale. PL/SQL di Oracle e il T-SQL di Microsoft SQL Server sono esempi di linguaggi con i quali è possibile scrivere stored procedure.

VISUAL STUDIO SERVER EXPOLER

L'operazione di creazione di una stored procedure dipende dal database sottostante. Generalmente esistono diversi metodi per inserire una procedura all'interno della struttura di un database. Molti database commerciali forniscono dei tool che, tra le tante funzionalità offerte, consentono anche la scrittura, la memorizzazione e l'esecuzione di stored procedure. Questi tool sono tipicamente applicazioni dotate di GUI che consentono di creare stored procedure in modo visuale oppure specificando il relativo comando pseudo SQL. E' possibile inoltre, creare stored procedure e funzioni direttamente da programma. In questo paragrafo analizzeremo un tool presente in Visual Studio .NET: il Server Explorer. Questo tool consente di interagire con la struttura di un database SQL Server. Per far partire Server Explorer, è necessario lanciare Visual Studio .NET è selezionare la voce di menù *Tools | Connect to a database ....*

Vantaggi

Una stored procedure consiste in un insieme di operazioni memorizzate ed eseguite sul server di database. Esse possono essere utilizzate per "spostare" parte della logica applicativa dal lato client a quello server. In questo modo, le applicazioni client risulteranno più leggere e il traffico di rete sensibilmente ridotto. Tutto questo porterà un vantaggio significativo in termini di prestazioni. Inoltre, il fatto che la logica applicativa sia centralizzata all'interno delle stored procedure, renderà il software più facile da "*manutenere*" e riutilizzare.

.NET e DB

Stored Procedure
con ADO.NET e SQL Server

Utile

La procedura presentata in questa pagina non serve assolutamente a nulla. Infatti opera sempre sullo stesso impiegato assegnandogli sempre lo stesso numero di telefono. Se avessimo la possibilità di specificare la matricola dell'impiegato ed il numero di telefono prima dell'esecuzione della query, allora tutto il discorso risulterebbe certamente più utile. Naturalmente questa possibilità esiste e può essere realizzata grazie ai parametri di input.

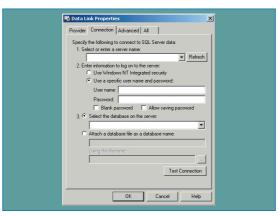



Fig. 1: **Dialog per la connessione ad un database.**

Apparirà una dialog che chiederà a quale database vogliamo connetterci (Fig. 1). Una volta ottenuta la connessione con il database desiderato, sulla sinistra apparirà la finestra relativa al Server Explorer (Fig. 2).

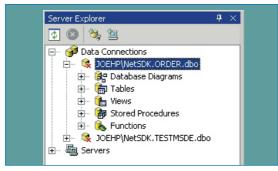



Fig. 2: **Il ServerExplorer di Visual Studio.**

Da adesso in poi è possibile interagire sia con la struttura sia con i dati. Per creare una stored procedure bisogna cliccare con il tasto destro del mouse sul nodo relativo alle stored procedure e scegliere *New Stored Procedure* dal pop-up menù che apparirà. A questo punto, nell'editor di Visual Studio .NET potrete vedere un *template* della stored procedure scritto in T-SQL (Fig. 3). Dopo aver scritto la procedura (vedremo in seguito numerosi esempi), sarà possibile salvarla, modificarla ed eseguirla; oppure effettuare altre operazioni previste nel pop-up menù relativo (attivabile con il tasto destro del mouse). Stesso discorso vale per le *funzioni* che sono semplicemente particolari tipi di stored procedure che restituiscono un valore. Se avete notato, nell'albero di Fig. 2, è presente anche il nodo relativo alle funzioni. Il modo di operare è pressoché identico a quello visto per le stored procedure: si possono creare, salvare, modificare ed eseguire. Attraverso il Server Exporer sarà possibile anche effettuare il debugging di stored procedure e funzioni, proprio come avviene con il codice scritto in Visual Basic, C++ o C#.

```
Module1.vb | dbo.Stipendi : S...HP\NetSDK.ORDER) | dbo.StoredPro...DK.TESTMSDE)*
CREATE PROCEDURE dbo.StoredProcedure1
/*
	(
		@parameter1 datatype = default value,
		@parameter2 datatype OUTPUT
	)
*/
AS
	/* SET NOCOUNT ON */
	RETURN
```

Fig. 3: **Template di una stored procedure.**

TIPI DI STORED PROCEDURE

Una stored procedure può essere classificata sia in base ai parametri che riceve in input, sia in base ai valori che restituisce in output. Di seguito veranno descritte le varie classificazioni. Le procedure *senza parametri* sono stored procedure che effettuano un lavoro "costante" e quindi non dipendono dall'entità che le invoca. La procedura *telefono*, di seguito riportata, è un esempio di stored procedure senza parametri che non fa altro che cambiare il numero di telefono di un determinato impiegato:

```
ALTER PROCEDURE dbo.telefono AS
UPDATE Impiegato SET telefono='06/8080'
WHERE matricola = '53-10642';
RETURN
```

Molte volte c'è la necessità di scrivere *procedure che restituiscono un valore*. In alcuni linguaggi esiste una differenza sostanziale tra procedure e funzioni. Le prime non restituiscono nulla, le seconde devono obbligatoriamente restituire un valore. In T-SQL esiste questa differenza. Il seguente esempio crea una funzione che restituisce il numero complessivo di impiegati:

```
CREATE FUNCTION dbo.conta_impiegati()
RETURNS INT AS
BEGIN
declare @tot int;
SELECT @tot = COUNT(*) FROM Impiegato;
RETURN @tot
END
```

La funzione esegue una *COUNT* sulla tabella *Impiegato* e inserisce il risultato nella variabile intera *tot* che sarà successivamente restituita. La funzione *conta_impigati* è una stored procedure a tutti gli effetti e come tale può essere richiamata direttamente dagli utenti del database o dalle applicazioni che vi si connettono. Molti DBMS consentono di utilizzare le funzioni anche all'interno di altre istruzioni SQL o stored procedure. Nell'esempio che segue saranno selezionate tutte le aziende che hanno un numero di impiegati superiore a quello restituito dalla stored procedure *conta_impiegati*:

```
SELECT * FROM Azienda
WHERE giorni > dbo.conta_impiegati()
```

Le stored procedure, siano esse funzioni o procedure, hanno la possibilità di accettare parametri *IN*, *OUT* e *IN OUT*. Un parametro *IN* è un valore che può essere letto e scritto dalla procedura, ma le alterazioni non sa-

ranno visibili all'esterno. Un parametro *OUT* ha la funzionalità inversa: è usato dalle stored procedure per assegnare valori che saranno visibili all'esterno della procedura. Un parametro *IN OUT* può essere usato indifferentemente come parametro *IN* o *OUT*. In una procedura con parametri *IN*, a differenza di quelle senza parametri, l'esecuzione sarà influenzata dai valori d'ingresso. Riscriviamo la procedura telefono, vista in precedenza, specificando come parametri di input l'impiegato ed il numero di telefono:

```
ALTER PROCEDURE dbo.cambia_telefono
@p_matricola char, @p_telefono char AS
UPDATE Impiegato SET telefono=@p_telefono
WHERE matricola = @p_matricola
RETURN
```

Adesso la procedura assume carattere generale, in altre parole può essere modificato il numero di telefono di qualsiasi impiegato. Esempi:

```
cambia_telefono('53-10642','06/677788')
cambia_telefono('53-10808','06/564343')
cambia_telefono('53-09887','06/458214')
```

L'esigenza di avere parametri *OUT* si ha quando una procedura deve restituire più di un valore. Abbiamo già visto che una funzione può restituire un valore. Esistono casi in cui questo può non essere sufficiente. Supponiamo di voler scrivere una procedura che restituisca il numero di impiegati che lavorano per un'azienda e il numero di quelli che hanno un'anzianità superiore ad n anni di servizio. Quello che serve in questo caso è una stored procedure con due parametri *OUT* ed uno *IN*. I parametri *OUT*, alla fine dell'esecuzione, conterranno rispettivamente il totale degli impiegati e quelli con anni di servizio superiore ad *p_n* (che rappresenta il parametro *IN*). La procedura può essere la seguente:

```
CREATE PROCEDURE dbo.servizio
@p_totale int OUTPUT, @p_servizio int OUTPUT, @p_n int AS
SELECT @p_totale = COUNT(*)
FROM Impiegato;
SELECT @p_servizio = COUNT(*)
FROM Impiegato
WHERE servizio >= @p_n;
RETURN
```

Come si può notare i parametri *p_totale* e *p_servizio* sono utilizzati per contenere il risultato delle due *COUNT*. In alcune circostanze un parametro può essere utilizzato sia come valore d'ingresso sia come valore d'uscita. Supponiamo di scrivere una procedura che, dato un numero in input, controlla se il valore massimo dell'attributo *servizio* della tabella *Impiegato* è maggiore di questo numero. Se questa situazione si verifica, al numero dato in input sarà assegnato il massimo, in caso contrario non succederà nulla. Se il numero in input è *p_max* allora alla fine dell'elaborazione esso potrà assumere o il valore del massimo tra gli attributi *servizio* oppure rimanere invariato. La stored procedure è la seguente:

```
CREATE PROCEDURE dbo.max_servizio
@p_max int OUTPUT AS
declare @tmp_max int
SELECT @tmp_max = MAX(servizio)
FROM Impiegato;
if (@tmp_max>@p_max) begin
  set @p_max = @tmp_max
end
RETURN
```

Supponiamo che l'impiegato più anziano abbia 43 anni di servizio allora:

```
max = 35
max_servizio(max)
print max <----- visualizzerà 43
```

invece

```
max = 46
max_servizio(max)
print max <----- visualizzerà 46
```

ESECUZIONE DI STORED PROCEDURE

Le applicazioni Visual Basic possono eseguire stored procedure e funzioni attraverso l'oggetto *Command* di *ADO .NET*. Il modo di operare è molto simile a quello usato per invocare una query. Attraverso la proprietà *CommandType* dell'oggetto *Command* è possibile comunicare al provider che si intende eseguire una stored procedure. Il valore da assegnare alla proprietà è *CommandType.StoredProcedure.* L'esecuzione di procedure senza parametri è sicuramente il caso più semplice. E' sufficiente infatti specificare solo il nome della procedura nella proprietà *Text* dell'oggetto *Command*, ed il gioco è fatto:

```
'Procedura senza parametri
Dim command As SqlCommand = New SqlCommand()
command.Connection = con
command.CommandText = "telefono"
command.CommandType = CommandType.StoredProcedure
command.ExecuteNonQuery()
```

L'esempio eseguirà la stored procedure *telefono* vista in precedenza. L'esecuzione di una stored procedure con parametri *IN* è molto simile a quella di una query parametrica. I parametri in ingresso dovranno essere impostati attraverso l'oggetto *SqlParameter*, la proprietà *Direction* indicherà che i parametri sono di tipo *IN* tramite il valore *ParameterDirection.Input.* Il seguente esempio invocherà la stored procedure *cambia_telefono* per l'impiegato 53-10642, dopo l'esecuzione il nuovo

.NET e DB

Stored Procedure
con ADO.NET e SQL Server

Bibliografia

• **LA PROGRAMMAZIONE DEI DATABASE CON VISUAL BASIC .NET**
G. Naccarato
G. Malorgio
(Apogeo)
2003

.NET e DB

Stored Procedure
con ADO.NET e SQL Server

numero di telefono sarà *06-7777:*

```
'Procedura con parametri IN
Dim command As SqlCommand = New SqlCommand()
command.Connection = con
command.CommandText = "cambia_telefono"
command.CommandType = CommandType.StoredProcedure
Dim pMatricola As SqlParameter = command.Parameters.
                              Add("@p_matricola", "53-10642")
pMatricola.Direction = ParameterDirection.Input
Dim pTelefono As SqlParameter = command.Parameters.
                                 Add("@p_telefono", "06/7777")
pTelefono.Direction = ParameterDirection.Input
command.ExecuteNonQuery()
```

Come si può notare, l'oggetto *Command* contiene la proprietà *Parameters* che rappresenta una *Collection* di *SqlParameter*. Prima di invocare la procedura è necessario aggiungere i parametri attraverso il metodo *Add*, passando il nome del parametro ed il relativo valore corrispondente; nel nostro caso:

```
command.Parameters.Add("@p_matricola", "53-10642")
command.Parameters.Add("@p_telefono", "06/7777")
```

A questo punto, il provider provvederà a sostituire ai parametri i valori opportuni prima di invocare la stored procedure. Attraverso il valore *ParameterDirection.Output* della proprietà *Direction* è possibile comunicare che un dato parametro è di output. Nell'esempio che segue eseguiremo la procedura servizio, che come ricorderete, prevede due parametri *OUT* (*p_totale* e *p_servizio*) ed uno *IN* (*p_n*):

```
Dim command As SqlCommand = New SqlCommand()
command.Connection = con
command.CommandText = "servizio"
command.CommandType = CommandType.StoredProcedure
Dim pTotale As SqlParameter = command.Parameters.
                             Add("@p_totale", SqlDbType.Int)
pTotale.Direction = ParameterDirection.Output
Dim pServizio As SqlParameter = command.Parameters.
                           Add("@p_servizio", SqlDbType.Int)
pServizio.Direction = ParameterDirection.Output
Dim pN As SqlParameter =command.Parameters.Add(
                                            "@p_n", 10)
pN.Direction = ParameterDirection.Input
command.ExecuteNonQuery()
Console.WriteLine ("Totale Impiegati: " & retParam.Value)
Console.WriteLine ("Con servizio superiore a " &
                     pN.Value & " anni: " & pServizio.Value)
```

Come si può notare, nel momento in cui si aggiunge un parametro di tipo *OUT*, è necessario anche specificarne il tipo. Nel nostro caso gli oggetti *SqlParameter pTotale* e *pServizio* saranno degli interi *(SqlDbType.Int)*. L'esempio visualizzerà i valori assegnati ai parametri *OUT* dalla stored procedure. Essi saranno reperiti attraverso la proprietà value di *SqlParameter*. Un possibile output potrebbe essere il seguente:

Totale Impiegati: 170
Con servizio superiore a 10 anni: 83

Per eseguire una stored procedure che prevede un parametro *IN OUT*, è sufficiente impostare la properietà *Direction* di *SqlParameter* al valore *ParameterDirection.InputOutput*. Nell'esempio che segue sarà eseguita la stored procedure *max_servizio*, con il parametro *p_max* di tipo *IN OUT*.

```
'Procedure con parametri IN OUTPUT
Dim command As SqlCommand = New SqlCommand()
command.Connection = con
command.CommandText = "max_servizio"
command.CommandType = CommandType.StoredProcedure
Dim pMax As SqlParameter = command.Parameters.Add(
                                          "@p_max", 46)
pMax.Direction = ParameterDirection.InputOutput
command.ExecuteNonQuery()
Console.WriteLine("p_max: " & pMax.Value)
```

Alla fine, sarà visualizzato l'eventuale nuovo valore di *p_max* .

ESECUZIONE DI FUNZIONI

Come abbiamo visto, le funzioni sono un particolare tipo di stored procedure che restituiscono un valore. Il valore di ritorno è visto da ADO .NET come un parametro la cui proprietà *Direction* assume valore *ParameterDirection.ReturnValue*.
Prima di eseguire una funzione, è necessario quindi aggiungere questo particolare tipo di parametro alla proprietà *Parameters* dell'oggetto *Command*. Ciò può essere fatto usando la funzione *Add* passando il nome del parametro ed il tipo. Il seguente frammento di codice eseguirà la funzione *conta_impiegati* e visualizzerà il risultato:

```
'Funzione
Dim command As SqlCommand = New SqlCommand()
command.Connection = con
command.CommandText = "conta_impiegati"
command.CommandType = CommandType.StoredProcedure
Dim retParam As SqlParameter = _
  command.Parameters.Add("ret", SqlDbType.Int)
retParam.Direction = ParameterDirection.ReturnValue
command.ExecuteNonQuery()
Console.WriteLine _
 ("Numero Impiegati: " & retParam.value)
```

Nell'esempio è stato assegnato *"ret"* come il nome del parametro di ritorno, questo valore è totalmente arbitrario e può essere scelto a discrezione dell'utente. Il frammento di codice visualizzerà il numero d'impiegati presenti nel sistema.

Giuseppe Naccarato

Adattamenti

In quest'articolo abbiamo introdotto le stored procedure ed abbiamo visto come utilizzarle con ADO .NET, Visual Basic ed SQL-Server. Tenete bene in mente che tutto ciò che è stato detto può facilmente adattato al linguaggio C# oppure ad un altro database quale Oracle o Sybase.

Programmazione Enterprise (parte seconda)

Applicazioni J2EE multicanale

J2EE

Accresciamo le possibilità della nostra rubrica aziendale con l'aggiunta dell'accesso alle informazioni via Web Service.

File sul CD
\soft\codice\mcapp_2.zip

File sul Web
www.ioprogrammo.net/files/72/mcapp_2.zip

Nell'articolo del mese scorso sono stati spiegati i concetti generali dei termini che costituiscono i mattoni della multicanalità: il dispositivo consumatore, il protocollo di erogazione ed il canale. È stata anche implementata una completa applicazione multicanale attraverso l'adozione della tecnologia J2EE ed utilizzando il Pattern MVC per il disegno dell'architettura. Le realizzazione è stata corredata di un componente Controller per il dispatching delle chiamate HTTP e di componenti di View e di Model che consentono l'erogazione del servizio attraverso i canali Web e Wap. Obiettivo del presente articolo è l'estensione di questa applicazione multicanale con l'aggiunta dell'erogazione di servizi sotto forma di Web Service attraverso il protocollo applicativo SOAP.

SCENARIO

L'oggetto di partenza è l'applicazione a tre livelli realizzata nell'articolo precedente, che abbiamo già detto essere stata progettata attraverso il pattern MVC e scritta in tecnologia J2EE.

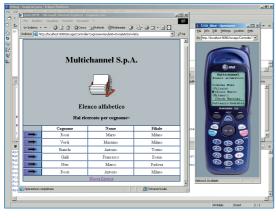
Fig. 1: L'applicazione multicanale completa.

L'applicazione gestisce nativamente il canale Web ed il canale Wap, come è visibile in Fig. 1, ed è estensibile, nel senso che è già predisposta all'erogazione degli stessi servizi su altri canali. L'applicazione esistente, dal punto di vista funzionale, implementa la completa gestione di una rubrica aziendale, in particolare sono state realizzate le funzionalità di:

- Motore di ricerca per cognome sul database della rubrica.
- Lista di tutti gli utenti compatibili con i criteri di ricerca inseriti.
- Dettaglio completo del singolo utente selezionato.

Estenderemo l'applicazione per rendere il motore di ricerca fruibile attraverso SOAP. ed utilizzeremo la componente Apache Axis per aggiungere alla nostra applicazione il canale Web Service.

APACHE AXIS

Axis è un SOAP Engine e consiste in un'implementazione Java della specifica SOAP. Si tratta della terza generazione di Apache SOAP ed è curata e rilasciata in modalità open source dalla Apache Software Foundation all'interno dell'ambizioso Apache XML Project. In questa versione Axis è stato profondamente ristrutturato e mentre prima si trattava di un semplice strato software in grado di renderci trasparente l'utilizzo di SOAP nello sviluppo di applicazioni scritte con tecnologia Java, adesso è un vero e proprio framework che ci permette di costruire, con estrema semplicità, tutto quello che ci può servire per rendere fruibili i nostri servizi sotto forma di Web Service: server, client, proxy, stub, skeleton e gateway da e verso il protocollo SOAP. Ci troviamo di fronte, quindi, ad un ambiente completo che garantisce l'interoperabilità con servizi che utilizzano tecnologie differenti, grazie al supporto completo e nativo per WSDL 1.1. Apache Axis è disponibile sul CD allegato, ed è disponibile per il download all'indirizzo: *http://ws.apache.org/axis/*. Per installare Axis sul nostro sistema è necessario scompattare il pacchetto completo xml-axis-10.zip ottenendo l'alberatura di Fig. 2. A questo punto sarà necessario copiare la directory webapps/axis all'interno della directory webapps del nostro application server, nel

J2EE

La complessa architettura di Java è divisa in tre piattaforme distinte.

Java 2 Standard Edition (J2SE) **definisce i costrutti del linguaggio, le librerie di utilizzo generale ed i suoi servizi di base.**

Java 2 Micro Edition (J2ME) **consente la produzione di applicazioni per dispositivi wireless mobili e per sistemi embedded.**

Java 2 Enterprirse Edition (J2EE) **è orientata allo sviluppo di applicazioni distribuite su sistemi Internet/Intranet e consente la produzione di componenti server di elevata complessità attraverso le tecnologie JSP, Servlet, EJB e Web Services. La piattaforma si completa attraverso l'adozione di consolidati standard di integrazione come JMS, JTS, JCA, JNDI.**

J2EE

Applicazioni
J2EE multicanale

Fig. 2: **L'alberatura completa di Apache Axis.**

caso specifico Tomcat 4. È necessario tener presente che Axis si porta dietro tutte le librerie di cui ha bisogno per funzionare ad eccezione del parser XML. E' possibile utilizzare qualsiasi parser a condizione che sia compatibile con le specifiche JAXP 1.1, tradizionalmente si utilizza Xerces, scaricabile liberamente da *http://xml.apache.org/dist/xerces-j/*.

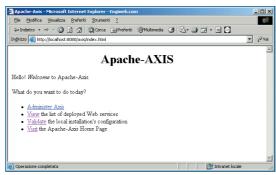



Fig. 3: **La pagina di benvenuto di Apache Axis, seguendo il link "validate" è possibile verificare l'installazione.**

Il file xerces.jar deve essere inserito nella directory *CATALINA_HOME/webapps/axis/WEB-INF/lib*. Una volta terminata la procedura di installazione è necessario riavviare il servizio di Tomcat e puntare il browser all'url: *http://localhost:8080 /axis/index.html*. Quello che si ottiene è mostrato in Fig. 3; se tutto è andato bene dovremmo aver installato correttamente Axis sul nostro sistema. Per verificare ulteriormente la qualità dell'installazione, è possibile seguire il link "Validate the local installation's configuration". Se si ottiene un messaggio del tipo: *"The core axis libraries are present."*, siamo sicuri di avere sul nostro sistema tutto ciò che ci serve per realizzare e testare Web Service con Axis. A questo punto siamo pronti per realizzare il nostro Web Service.

Il pattern MVC

Uno degli obiettivi del paradigma *Object Oriented* è la possibilità di realizzare classi che possano vivere autonomamente in qualsiasi contesto e non solo in quello nel quale vengono realizzate.
Questo implica che le componenti di un'applicazione debbano essere sufficientemente separate tra loro e che possano essere sostituite con altre implementate diversamente, a condizione che ne rispettino l'interfaccia.
Per realizzare applicazioni che soddisfino questi requisiti, si utilizza il pattern MVC che consente di separare tra loro le componenti applicative: il Model che implementa la funzionalità di business, la componente di View che implementa la logica di presentazione ed il Controller che implementa la logica di controllo.

IL SERVIZIO

Il nostro obiettivo è rendere fruibile, sotto forma di Web Service, un servizio che riceva una chiave di ricerca e ci restituisca una lista di tutti i nominativi presenti nella rubrica, il cui cognome contiene la chiave di ricerca stessa. Per prima cosa dobbiamo creare la struttura del servizio e, poiché si utilizzerà Axis per rendere il servizio fruibile come Web Service, dovremo creare un'interfaccia con il metodo da rendere disponibile sul canale SOAP e la classe che lo implementa. Questa struttura è visibile in Fig. 4.

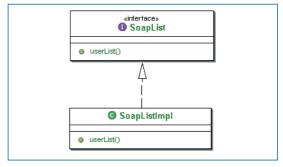



Fig. 4: **Class diagram dell'interfaccia del servizio e della sua implementazione.**

L'interfaccia è contenuta nel file *SoapList.java* ed ha il seguente contenuto:

```
package com.mcapp.services;
public interface SoapList {
    public String userList(String cognome);
}
```

mentre la classe che implementa il servizio è *SoapListImpl.java*, eccone il codice:

```
package com.mcapp.services;
import java.sql.*;
public class SoapListImpl implements SoapList
{
    public String userList(String cognome)
    {
        String output = "";
        try
        {
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          Connection con = DriverManager.getConnection
                                    ("jdbc:odbc:mcapp");
            Statement stmt = con.createStatement();
            String qq ="SELECT id, cognome, nome, filiale
                  FROM rubrica where cognome LIKE '%"
                                  + cognome + "%'";
            ResultSet rs = stmt.executeQuery(qq);
            while (rs.next())
            {
                for (int col = 2;
                    col <= rs.getMetaData(
                              ).getColumnCount(); col++)
                {
                    output += rs.getString(col);
                    output += ", ";
                }
                output += "\n";
            }
        }
        catch (Exception e)
        {}
```

```
        return output;
    }
    public static void main(String[] args)
    {
        SoapListImpl c = new SoapListImpl();
        System.out.println(c.userList("a"));
    }
}
```

Si tratta, come si può vedere, di un servizio che riceve in input una stringa di criteri ed esegue una query su database ricercando tutti gli utenti censiti nella rubrica aziendale e che contengono all'interno del cognome la stringa passata come parametro. Come si può notare dalla dichiarazione del package, la classe e la sua interfaccia sono contenute, nell'alberatura dell'applicazione, all'interno della directory src*com\mcapp\services*. All'interno della classe che implementa il servizio è presente anche un metodo main. Questo si spiega in quanto, quando si scrive un servizio, è necessario testarlo localmente con dei dati di prova prima di renderlo disponibile all'esterno. Se proviamo, dall'interfaccia di Eclipse, ad eseguire il servizio, come visibile in Fig. 5, otterremo nella console i due record che soddisfano il criterio che il main gli ha passato, possiamo pertanto affermare che il servizio funziona e, se vogliamo, possiamo rimuovere il metodo main dalla classe.

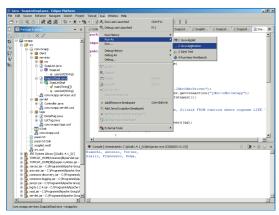



Fig. 5: **Esecuzione del servizio locale nell'interfaccia di Eclipse.**

LA STRUTTURA DEL WEB SERVICE

Per trasformare una classe che implementa un servizio in un vero e proprio Web Service, abbiamo a disposizione le funzionalità di Axis che ci consentono di costruire, in maniera semiautomatica, tutta l'infrastruttura software necessaria al nostro scopo. Per prima cosa dobbiamo ottenere il file di descrizione del servizio, il file WSDL, e per averlo utilizziamo l'utility Java2WSDL contenuta nel package di Axis. Per fare questo dobbiamo recarci, nel prompt dei comandi, nella directory che rappresenta la root dei nostri sorgenti; nel nostro caso si tratta della directory *C:\Progetti\mcapp\src*. Da questa posizione dobbiamo eseguire il seguente comando:

```
java org.apache.axis.wsdl.Java2WSDL -o soaplist.wsdl -
        l"http://localhost:8080/axis/services/soaplist" -n
                urn:soaplist -p"soaplist" urn:soaplist
                        com.mcapp.services.SoapList
```

con il quale chiediamo al framework di generare per noi il file *soaplist.wsdl* a partire dall'interfaccia *com.mcapp.services.SoapList*, per un Web Service che sarà disponibile agli url ed urn indicati. Se non ci sono errori, il più tipico dei quali è dovuto alla dimenticanza di includere nel classpath tutti i file .jar contenuti nella directory lib di Axis, verrà generato il file *soaplist.wsdl* che descrive nei minimi dettagli il nostro servizio web. A questo punto siamo pronti per generare, a partire da questo file WSDL, l'infrastruttura software necessaria per rendere fruibile il nostro servizio sul canale SOAP. L'operazione consiste nel generare numerosi file in grado di far comunicare il nostro servizio con il framework di Axis:

```
SoapList.java
SoapListService.java
SoapListServiceLocator.java
SoaplistSoapBindingImpl.java
SoaplistSoapBindingStub.java
```

E la coppia di file che si occupa di configurare il deploy e l'undeploy del servizio:

```
deploy.wsdd
undeploy.wsdd
```

In realtà a tutto questo pensa Axis, il nostro compito sarà soltanto quello di utilizzare un apposito tool presente nel framework; penserà lui a generare il tutto. Rechiamoci nuovamente nella directory, che è la nostra root per i sorgenti, ed utilizziamo il tool con questo comando:

```
java org.apache.axis.wsdl.WSDL2Java -o . -d Session
        -s -p com.mcapp.services.ws soaplist.wsdl
```

con il quale chiediamo al framework di generare, a partire dal file *soaplist.wsdl* che avevamo creato in precedenza, tutta l'infrastruttura che software per gestire un servizio con *scope=Session*. La generazione dei file è immediata ed infatti troveremo nella directory *…/services/ws* tutti i file che servono per gestire pubblicare il servizio come Web Service.

L'IMPLEMENTAZIONE

Tra i vari file generati automaticamente dal tool c'è anche *SoaplistSoapBindingImpl* che contiene l'implementazione di un ipotetico servizio, con le caratteristiche descritte nel file WSDL che abbiamo dato in pasto al tool. In realtà, però, ci aspetta una sorpresa; andiamo

J2EE

Applicazioni
J2EE multicanale

SOAP

SOAP è un protocollo applicativo che fornisce le regole per permettere ad applicazioni distribuite di scambiarsi informazioni e di richiedere l'esecuzione di servizi remoti.
Le applicazioni si scambiano messaggi attraverso un pacchetto informativo che prende il nome di payload, una struttura XML complessa.
Non esistono restrizioni per quel che riguarda il protocollo di trasporto utilizzato, l'unico vincolo è che sia in grado di trasportare il semplice testo.
In realtà, sebbene non esistano, controindicazioni di sorta verso altro protocolli, si tende a privilegiare HTTP per la sua ampia distribuzione e per le caratteristiche di request/response simili a quelle di SOAP.

J2EE

Applicazioni
J2EE multicanale

a vedere il contenuto di questo file:

```
/**
 * SoaplistSoapBindingImpl.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis WSDL2Java emitter.
 */
package com.mcapp.services.ws;
public class SoaplistSoapBindingImpl implements
                        com.mcapp.services.ws.SoapList
{
    public java.lang.String userList(java.lang.String in0)
        throws java.rmi.RemoteException
    {
        return null;
    }
}
```

Possiamo vedere che il metodo userList, cioè quello che viene in effetti chiamato dai client remoti via SOAP, restituisce sempre il valore null; perché? La risposta è semplice e deriva dal processo che abbiamo seguito per ottenere questo file. A partire dal file di implementazione originale, infatti, è stato ottenuto il file WSDL di descrizione del servizio, che però non contiene informazioni riguardo l'implementazione. A partire da quest'ultimo poi è stato generato, tra gli altri, questo file che rappresenta l'implementazione del Web Service, ma il tool non ha potuto condirlo con del codice reale, in quanto non è a conoscenza della reale implementazione; infatti, ha ricevuto in input soltanto la descrizione del servizio nel file WSDL. Tocca a noi quindi fare questa piccola parte di lavoro e modificare il codice dell'implementazione del Web Service, per fare in modo che utilizzi la stessa implementazione del servizio locale, utilizzandone di fatto un'istanza. Ecco il codice modificato:

```
package com.mcapp.services.ws;
import com.mcapp.services.SoapListImpl;
public class SoaplistSoapBindingImpl implements
                        com.mcapp.services.ws.SoapList
{
        SoapListImpl lista = new SoapListImpl();
    public java.lang.String userList(java.lang.String in0)
        throws java.rmi.RemoteException
    { return lista.userList(in0);}
}
```

Per prima cosa, il nuovo codice esegue l'import della classe che contiene l'implementazione del servizio, poi ne crea un'istanza e la utilizza per generare il valore che sarà restituito al chiamante. Adesso, attraverso i seguenti procedimenti, il cerchio si è chiuso:

- Costruzione di un servizio locale con tanto di test personalizzato.
- Generazione del file WSDL di descrizione del servizio attraverso il tool *Java2Wsdl*.
- Generazione dell'infrastruttura software per un generico servizio, descrivibile con quel file di descrizione utilizzando il tool *Wsdl2Java*.
- Personalizzazione del codice generato automaticamente per consentire l'utilizzo remoto dell'implementazione del servizio locale.

Naturalmente, i file generati dai tool devono essere compilati, a meno di utilizzare Eclipse che compila automaticamente i sorgenti quando vengono salvati. Una vista d'insieme del progetto all'interno dell'ambiente di Eclipse è visibile in Fig. 6.

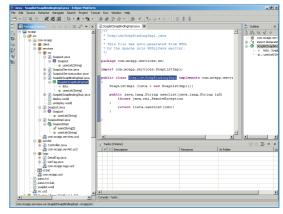



Fig. 6: **Vista d'insieme del progetto in Eclipse.**

A questo punto non ci resta che effettuare la pubblicazione del servizio.

IL DEPLOY

Per fare la pubblicazione del servizio, è necessario innanzi tutto creare in file jar contenente tutti i file del servizio reale e dell'infrastruttura di pubblicazione. Per ottenere questo risultato è necessario recarsi nuovamente nella directory root dei sorgenti ed eseguire il comando:

```
jar cfv soaplist.jar com/mcapp/services/*.class
                        com/mcapp/services/ws/*.class
```

Il file *soaplist.jar* generato, per essere visibile da Axis, dovrà essere collocato nella directory *CATALINA_HOME/webapps/axis/WEB-INF/lib*. A questo punto possiamo far ripartire il servizio di Tomcat e, dopo esserci spostati nella dirertory *…/services/ws* contenente il file *deploy.wsdd*, possiamo effettuare il deploy attraverso il comando seguente:

```
java org.apache.axis.client.AdminClient deploy.wsdd
```

Se non ci sono errori, il tool risponderà con il seguente output:

- Processing file deploy.wsdd
- <Admin>Done processing</Admin>

WSDL

WSDL è una specifica del W3C

www.w3c.org/TR/wsdl

che ha l'obiettivo di fornire la descrizione di un Web Service utilizzando un'applicazione di XML. Poiché i Web Service sono per definizione distribuiti su piattaforme differenti e consumati da client che a priori non sono noti, è bene definire un file WSDL per ogni servizio che si produce, in modo da disaccoppiare il servizio dal client che cerca di utilizzarlo. In questo modo il client non utilizza direttamente il Web Service, ma passa attraverso la sua descrizione in WSDL per capire come fare ad utilizzare il servizio.

ed il nostro Web Service sarà pronto a rispondere all'url che abbiamo indicato durante la produzione del file WSDL. Se vogliamo verificare che il nostro Web Service sia stato pubblicato correttamente, all'interno del framework di Axis, possiamo andare all'url *http://localhost:8080/axis/index.html* e seguire il link "View the list of deployed Web services" che mostra la lista di tutti i servizi attualmente disponibili. In Fig. 7 viene mostrata la lista dei servizi nella quale, come si può vedere, è presente anche il nostro soaplist.

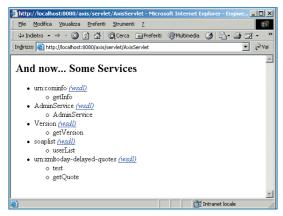



Fig. 7: **Lista dei servizi pubblicati all'interno del framework Axis; è presente anche il servizio soaplist.**

IL CLIENT

Adesso che abbiamo pubblicato il servizio, possiamo costruire un consumatore in grado di utilizzarlo. Per prima cosa creiamo, nell'alberatura dei nostri sorgenti, la directory *src\com\mcapp\client* che conterrà il nostro client di esempio. All'interno di questa directory creiamo la classe *SoapListTester* che implementa il nostro client; eccone i codice:

```
package com.mcapp.client;
public class SoapListTester
{
    public static void main(String[] args) throws Exception
    {
        if (args == null || args.length != 1)
        {
            System.err.println("Utilizzo corretto: ");
            System.err.println("java
                com.mcapp.client.SoapListTester stringa ");
            return;
        }
        String criterio = new String(args[0]);
        com.mcapp.services.ws.SoapListService service =
            new com.mcapp.services.ws.
                                SoapListServiceLocator();
        com.mcapp.services.ws.SoapList lista =
                                service.getsoaplist();
        System.out.println("La ricerca con il criterio=
                " + criterio + " ha prodotto il seguente
            risultato:\n " + lista.userList(criterio)); }
}
```

Come si può vedere, la chiamata al Web Service assomiglia tanto ad una chiamata ad in servizio locale, infatti il client non deve far altro che istanziare il servizio attraverso l'istruzione:

```
com.mcapp.services.ws.SoapListService service =
new com.mcapp.services.ws.SoapListServiceLocator();
```

poi deve istanziare il metodo remoto attraverso l'istruzione:

```
com.mcapp.services.ws.SoapList lista =
                                service.getsoaplist();
```

e deve eseguirlo passandogli come parametro il criterio di ricerca:

```
lista.userList(criterio)
```

Il risultato dell'esecuzione non è altro che il valore ritornato dall'esecuzione del metodo remoto. Questo dato viene infine stampato a video. Per verificare il funzionamento dell'applicazione di test, dobbiamo portarci nella posizione del file-system che rappresenta la root dei nostri sorgenti ed eseguire il comando:

```
java com.mcapp.client.SoapListTester criterio
```

In Fig. 8 è visibile l'utilizzo dell'applicazione di test.

```
C:\Progetti\mcapp\src>java com.mcapp.client.SoapListTester
Utilizzo corretto:
java com.mcapp.client.SoapListTester stringa

C:\Progetti\mcapp\src>java com.mcapp.client.SoapListTester rossi
La ricerca con il criterio= rossi ha prodotto il seguente risultato:
 Rossi, Mario, Milano,
Rossi, Antonio, Milano,

C:\Progetti\mcapp\src>java com.mcapp.client.SoapListTester o
La ricerca con il criterio= o ha prodotto il seguente risultato:
 Rossi, Mario, Milano,
Rossi, Antonio, Milano,

C:\Progetti\mcapp\src>_
```

Fig. 8: **Esecuzione del client in una finestra di command. Il consumatore utilizza il Web Service per mostrare i dati all'utente.**

CONCLUSIONI

In questo articolo abbiamo esteso l'applicazione multicanale mcapp per fare in modo che alcuni servizi potessero essere erogati anche attraverso l'utilizzo del canale SOAP. Per raggiungere questo obiettivo è stato utilizzato il framework open source Apache Axis, che permette di concentrarsi sulla logica del servizio da realizzare, senza preoccuparsi troppo della semantica della comunicazione tra client e server attraverso il protocollo applicativo SOAP.
Mediante i tool messi a disposizione del framework, è stato generato il file WSDL di descrizione del servizio, a partire da questo si è generata automaticamente tutta l'infrastruttura software in grado di far comunicare tra loro i client consumatori con il servizio stesso. I servizi della nostra applicazione multicanale sono erogabili, adesso, anche sotto forma di Web Service.

Massimo Canducci

J2EE

Applicazioni
J2EE multicanale

Apache Axis

Apache Axis è l'evoluzione di Apache SOAP 2.2, da cui eredita le caratteristiche di base eliminando i difetti del predecessore:

- **È perfettamente compatibile con Microsoft SOAP Toolkit.**
- **Supporta le specifiche WSDL 1.1.**

Inoltre, sono previste le seguenti novità:

- **Supporto parziale delle specifiche SOAP 1.2.**
- **Supporto per la pubblicazione automatica dei servizi (Java Web Service).**
- **Generazione automatica (direttamente da un browser Internet) del documento WSDL per un servizio pubblicato.**
- **Tool WSDL2Java e Java2WSDL.**

Java

☑ Java per principianti.

Un po' di logica

Questo mese facciamo il nostro terzo passo sulla luminosa via di Java, riprendendo il discorso da dove lo avevamo interrotto. Parleremo dei tipi del linguaggio, e in particolare di un tipo che si chiama boolean. Che ci spalancherà un mondo cristallino di logica, e ci permetterà, finalmente, di scrivere programmi un po' più interessanti delle solite stampe.

File sul CD
\soft\codice\java3.zip

Sul CD allegato a questo numero di IoProgrammo troverai un file ZIP che contiene tutto il codice degli esempi di questo mese.

File sul Web
www.ioprogrammo.net/files/72/java3.zip

Tutto ha un segno
A differenza di quanto accade in C, in Java non esistono tipi numerici "senza segno". Sia i tipi interi come *int* che quelli in virgola mobile come *double* possono sempre rappresentare sia quantità positive che quantità negative.

Come al solito, non perdiamo tempo e stabiliamo quali sono gli obiettivi per questa lezione.

Obiettivi di questa lezione:

- Farai la conoscenza dei *tipi primitivi* del linguaggio.
- Imparerai cos'è la priorità degli operatori e come gestirla.
- Farai la conoscenza del *tipo boolean* e degli operatori logici.
- Incontrerai la tua prima istruzione per controllare la logica del programma.

DUE TIPI DI TIPI

Il mese scorso ho scritto che in Java esiste un numero infinito di tipi, perché il programmatore può creare i propri tipi da sé. Questa è una caratteristica della programmazione object-oriented, della quale parleremo molto tra qualche mese. Per ora ci interessano solo i tipi predefiniti dal linguaggio Java, che non hanno niente a che vedere con la programmazione ad oggetti. Questi tipi derivano direttamente da quelli del linguaggio C, e ci permettono di definire delle semplici variabili come abbiamo fatto il mese scorso:

```
int x;
```

Questa istruzione definisce una variabile *x*, e specifica che il valore di *x* deve essere di tipo *int*. Il tipo *int* è il più comune tra i numerosi tipi interi di Java: un intero rappresentato in notazione a 32 bit con segno. Se non sai cosa significa, ti basta sapere che un *int* può contenere valori che vanno da -2^{31} a 2^{31}-1, cioè all'incirca da meno due miliardi a più due miliardi. Se provi a mettere in un *int* un valore più grande (o un valore non intero), aspettati un errore durante l'esecuzione del programma. Se vuoi manipolare numeri interi più grandi puoi usare un *long*, che con i suoi 64 bit permette di rappresentare numeri fino a quasi 10 miliardi di miliardi. Se invece vuoi rappresentare numeri più piccoli e sei particolarmente avaro di memoria puoi fare ricorso al tipo *byte* (che appunto contiene un byte, e quindi un numero da –128 a +127) o al tipo *short* (16 bit, da –32768 a +32767). In realtà di solito la memoria non è un grande problema, e chi usa i tipi interi più piccoli lo fa più per comunicare le proprie intenzioni al lettore del codice che per risparmiare RAM. Come facciamo a rappresentare numeri decimali? Hai già fatto la conoscenza del tipo *double*, un numero in rappresentazione a 64 bit in "virgola mobile". Questa rappresentazione è abbastanza precisa per la maggior parte degli scopi, a meno che tu non debba scrivere del software matematico o finanziario. Conviene quasi sempre usare dei *double*, ma se sai per certo che la variabile conterrà solo numeri abbastanza piccoli e con poche cifre dopo la virgola puoi ripiegare sul tipo *float*, che è più piccolo (virgola mobile a 32 bit).

Un tipo un po' diverso dagli altri è *char*, che contiene un carattere Unicode. Le costanti carattere si mettono tra singoli apici. Ad esempio:

```
char c = 'X';
```

Nota che c'è differenza tra 'X' (il carattere X) e "X" (una stringa contenente solo un carattere X).

SIA DETTO TRA PARENTESI

Mettiamo da parte i tipi per un po', e torniamo agli operatori. Secondo te, qual è l'output di questo programma?

```
double x = 3 + 2 * 6;
System.out.println(x);
```

Se hai risposto 15, magari solo perché ti sei fidato dei tuoi ricordi scolastici, allora complimenti: ci hai preso. Se hai risposto 30, allora devi fare la conoscenza di un concetto importante: quello di *precedenza degli operatori*. Nelle espressioni, gli operatori non vengono sempre valutati da sinistra verso destra come si potrebbe pensare (in questo caso il calcolo sarebbe: "3 più 2, e il risultato moltiplicato per 6"). Esistono invece alcuni operatori che vengono valutati prima degli altri. Ad esempio, la moltiplicazione e la divisione hanno la precedenza sulla somma e la sottrazione. Quindi l'espressione del nostro esempio viene valutata come: "3 più la moltiplicazione di 2 per 6". Per cambiare la precedenza nelle espressioni possiamo usare le parentesi tonde. Quello che è tra parentesi tonde viene valutato prima. Ad esempio:

```
double x = (3 + 2) * 6;  // quanto vale x?
System.out.println(x);
```

Ricorda che tutto quello che segue la doppia barra è un commento, e che le espressioni tra parentesi vengono sempre valutate prima delle altre. In questo caso l'intera espressione viene valutata proprio come: "3 più 2, e il risultato moltiplicato per 6". Nelle espressioni più complicate ti capiterà di usare più livelli di parentesi tonde. Ti propongo un esercizio (la soluzione è verso la fine dell'articolo).

Esercizio 1: Prova a indovinare l'output di questo programma. Poi fallo girare e verifica se il tuo calcolo era giusto.

```
class Parentesi {
  /*  Questo esercizio mostra come si usano
      le parentesi per controllare la priorita'
      degli operatori. */
   public static void main(String[] args) {
      double x = (2 + ((3 + 3) * (2 + 4))) / 2;
      System.out.println("x = " + x);
  }
}
```

In questo codice vediamo anche un secondo modo di scrivere i commenti in Java. Tutto quello che si trova tra la sequenza di caratteri /* e la sequenza */ è un commento. A differenza del commento con "doppia barra", che termina sempre alla fine della riga, il commento con "barra e asterisco" può occupare tutte le righe che vogliamo.

LA VERITÀ (E IL SUO CONTRARIO)

Torniamo ai tipi. Ce ne resta solo uno, ma molto importante: il tipo *boolean*. Nella cosiddetta *logica booleana* le variabili possono assumere solo due valori: vero *(true)* e falso *(false)*.
Ad esempio:

```
boolean variabileLogica = true;
boolean altraVariabileLogica = false;
```

Se provi ad assegnare ad una variabile booleana un qualsiasi valore che non sia *true* o *false*, il compilatore ti dà un errore. Lo stesso succede se assegni un valore booleano ad una variabile di qualsiasi tipo che non sia *boolean*.

MELE E ARANCE

Come facciamo ad ottenere un valore booleano? Il primo modo: possiamo scriverlo come una costante (*true* o *false*). Il secondo modo: possiamo usare un operatore relazione oppure un *operatore logico*. Quelli relazionali e logici sono operatori simili agli altri, ma anziché dare un risultato numerico danno un risultato booleano.
Gli operatori relazionali sono quelli che confrontano due espressioni (ricordati che un'espressione può essere una cosa semplice come una costante o una variabile, oppure un calcolo molto complicato). L' operatore relazionale più semplice è quello che verifica se i valori delle due espressioni sono uguali. Si scrive con un doppio segno di uguale (==) e si chiama *operatore di uguaglianza*. Ecco un esempio:

```
int x = 1;
System.out.println(x == 2);
```

Questo codice stampa sullo schermo la parola "false". Nota che possiamo stampare un booleano esattamente come una stringa o come un valore numerico. Il doppio segno di uguale può sembrare una scelta strana, ma è indispensabile per distinguere questo operatore da quello di assegnamento, che si scrive come un solo segno di uguale. Ricorda che i due operatori sono completamente diversi.
L'assegnamento mette il valore dell'espressione sulla destra nella variabile sulla sinistra; il confronto è invece un'espressione booleana che vale *true* se l'espressione sulla destra e quella sulla sinistra sono uguali, e *false* in caso contrario.

Esercizio 2: Guarda il codice che segue. E' quasi identico al precedente, tranne che per una piccola differenza.

```
int x = 1;
System.out.println(x = 2);
```

Riesci a indovinare cosa stampa questo codice?
La domanda era un po' perfida, ma magari hai indovinato da solo. Il codice che hai appena visto stampa il valore 2. Infatti, come per tutti gli operatori, anche l'assegnamento ha un valore: è il valore dell'espressione

Tutte tonde

A differenza di quanto facevamo sui banchi di scuola, nella programmazione non possiamo usare le parentesi tonde, quadre e graffe per distinguere tre "livelli" di priorità nella valutazione delle espressioni. Si usano solo le tonde, eventualmente con più livelli:

a + (b + (c + d) + (e + f));

Questa espressione calcola prima *(c + d)* e *(e + f);* poi somma *b* con *(c + d)* e con *(e + f);* infine somma *a* con il risultato dell'ultima somma.

Java

sulla destra (in questo caso la costante numerica 2). Quindi la seconda riga di questo codice assegna alla variabile *x* il valore 2, e subito dopo stampa il valore assegnato sullo schermo. Ti spiego queste cose perché è abbastanza facile confondere i simboli = e ==, quindi è bene che tu sia preparato a riconoscere un eventuale errore. Un altro operatore relazionale che usa un simbolo strano è l'operatore di *non uguaglianza*, che si scrive con un punto esclamativo seguito da un uguale *(!=)*. Questo operatore è l'inverso dell'uguaglianza, e restituisce *true* solo se i due operandi sono diversi. Infine ci sono gli altri operatori di confronto: maggiore (>), minore *(<)*, maggiore o uguale *(>=)* e minore o uguale *(<=)*. Se hai difficoltà a riconoscere la direzione delle freccette, usa questo trucco mnemonico: la parte "piccola" della freccia (la punta) sta sempre dalla parte dell'operando più piccolo.

LOGICA PER LE TUE MENINGI

Come gli operatori relazionali, anche gli operatori logici restituiscono sempre un valore booleano. La differenza sta nel fatto che mentre gli operandi di un operatore relazionale possono avere qualsiasi tipo, gli operandi di un operatore logico devono, a loro volta, essere booleani. Il più semplice operatore logico è il *not*, che si scrive con un singolo punto esclamativo. L'operatore *!* ha un solo operando booleano, e semplicemente ne inverte il valore. Se l'operando è *true*, il *not* restituisce *false*, e viceversa. Esempio:

```
boolean b = true;
System.out.println(!b); // stampa false
int x = 1;
System.out.println(!(x == 2)); // stampa true
System.out.println(x != 2); // identica alla riga
                                        precedente
```

Un operatore logico importantissimo è l'*and*, che si indica con una doppia "e commerciale": *&&*. L'operatore *&&* prende due operandi booleani, e restituisce *true* solo se *entrambi* gli operandi sono *true*. Il suo compagno è l'operatore *or*, che si indica con una doppia barra *(||)* e restituisce *true* solo se *almeno uno* tra i due operandi è *true*. Ecco un programmino che dimostra tutti questi operatori:

```
class OperatoriLogici {
  public static void main(String[] args) {
    int x = 1;
    int y = 2;
    System.out.println("x = " + x);
    System.out.println("y = " + y);
    System.out.println("x uguale a y: " + (x == y));
    System.out.println("x diverso da y: " + (x != y));
    System.out.println("x minore di y: " + (x < y));
    System.out.println("x minore o uguale a y: " +
                                              (x <= y));
    System.out.println("x maggiore di y: " + (x > y));
    System.out.println("x maggiore o uguale a y: " +
                                              (x >= y));
  }
}
```

Nota che nelle stampe i valori booleani vengono convertiti in stringhe dall'operatore di concatenazione (+), proprio come succede per i valori numerici.

Esercizio 3: Nel codice qui sopra, prova a scrivere accanto a ciascuna stampa il valore (*true* o *false*) che ne risulterà.

Ecco l'output del programma *OperatoriBooleani*:

```
x = 1
y = 2
x uguale a y: false
x diverso da y: true
x minore di y: true
x minore o uguale a y: true
x maggiore di y: false
x maggiore o uguale a y: false
NOT x uguale a 1: false
x uguale a 1 AND y uguale a 2: true
x uguale a 1 OR y uguale a 3: true
x uguale a 1 AND y uguale a 1: false
x uguale a 2 OR (x uguale a 1 AND y uguale a 1): false
```

L'espressione più complicata è l'ultima. L'operatore *&&* viene eseguito per primo per via delle parentesi. Restituisce *false*, perché uno dei sue due operandi è *true* e l'altro è *false*. Quindi l'operatore || restituisce *false*, perché entrambi i suoi operandi sono *false*.

NELLA VITA BISOGNA FARE DELLE SCELTE

Ora siamo arrivati ad un punto di svolta nel nostro corso. Finora abbiamo scritto solo programmi che fanno di conto e stampano i risultati sullo schermo. E' venuto il momento di divertirci un po' con le prime *istruzioni di controllo* - quelle che fanno prendere delle decisioni al nostro codice. La più semplice e famosa è l'istruzione *if*:

```
if(condizione)
  istruzione1;
else
  istruzione2;
```

Questa istruzione dice: *se (if)* la *condizione* è vera, allora esegui l'*istruzione1*, *altrimenti (else)* esegui l'*istruzione2*. La condizione deve essere una qualsiasi espressione booleana. Potrebbe essere una espressione relazionale complicata, oppure una semplice variabile *boolean*. A Java non importa quello che c'è tra le due parentesi dell'*if*, purché valga *true* oppure *false*. Vediamo un bloc-

Sii bilanciato

Un classico errore di compilazione che è difficile individuare nel codice è quello che riguarda le *"parentesi sbilanciate"*. Ad esempio, questa espressione è scritta male:

a + (b + (c + d) + (e + f);

Ciascuna parentesi che apri deve essere chiusa una ed una sola volta, se non vuoi far arrabbiare il compilatore.

co *"if-else"* in azione. Il codice che segue controlla se una variabile di nome *numero* è pari o dispari. Riesci a capire come funziona?

```
if((numero % 2) == 0)
  System.out.println("Il numero e' pari");
else
  System.out.println("Il numero e' dispari");
```

La condizione dell'*if* è un'espressione che fa due cose:

- calcola il "modulo 2" della variabile, cioè la divide per due e prende il resto della divisione;
- confronta questo resto con 0.

Per fare il confronto ho usato l'operatore di uguaglianza. Se la variabile *numero* ha un valore pari, il resto della divisione è 0 e tutta l'espressione vale *true*; se ha un valore dispari, il resto è 1 e l'espressione vale *false*. Nel primo caso viene eseguita l'istruzione dopo l'*if*, altrimenti viene eseguita l'istruzione dopo l'*else*. L'istruzione *else* è facoltativa. Ad esempio, il codice che segue somma due variabili. Se la variabile *somma* è maggiore di 100 la "taglia" al valore 100, altrimenti non fa niente:

```
int somma = primoOperando + secondoOperando;
if(somma > 100)
  somma = 100;
```

Questo esercizio mette insieme diverse cose che abbiamo spiegato in questo articolo:

Esercizio 5:

```
class Coffee {
  public static void main(String[] args) {
    boolean hoDelDecaffeinatoInCasa = false;
    boolean miSonoSvegliatoDaPoco = false;
    boolean hoVogliaDiBereUnCaffe = true;
    if(/* metti qui la condizione */)
      System.out.println("Bevi pure tranquillo");
    else
      System.out.println("Meglio evitare il caffe'");
  }
}
```

Completa questo programma scrivendo la condizione dell'*if*. Le regole sono: se non ho voglia di caffè non ne bevo; altrimenti bevo solo se mi sono svegliato da poco, oppure se ho del decaffeinato in casa. Per esprimere queste condizioni basta un'unica espressione booleana con un *and* e un *or*. Come al solito troverai la soluzione sul CD. Prova a cambiare il valore delle tre variabili booleane per vedere come cambia l'output del programma. Ora che hai iniziato a capire come far prendere delle decisioni ai tuoi programmi, non ti fermerà più nessuno. L'istruzione *if* è solo l'inizio. Il mese venturo imparerai altri modi di controllare l'esecuzione del programma, e farai conoscenza con un altro concetto fondamentale della programmazione. Prima di salutarti, però, devo mantenere una promessa e darti la soluzione dell'*Esercizio 1*.

UN'ULTIMA COSA

*double x = (2 + ((3 + 3) * (2 + 4))) / 2;*

Il problema era: quanto vale *x*? Il calcolo viene fatto come: 3 più 3, moltiplicato per 2 più 4, il tutto sommato con 2 e infine diviso per 2 – quindi 19. Ma dato che il risultato è un *double*, sullo schermo verrà stampato il valore decimale *19.0*.
Nota che non tutte le parentesi in questa espressione sono necessarie. Una regola importante è quella di usare sempre le parentesi tonde nelle espressioni non banali, anche quando non sono indispensabili. Il computer conosce la priorità di tutti gli operatori, ma gli esseri umani che leggono il codice potrebbero non ricordarsela. Quindi è sempre meglio usare le parentesi.

Esercizio 6: L'espressione che viene assegnata a *x* nell'*Esercizio 1* contiene quattro coppie di parentesi, e una sola tra queste è superflua: potremmo cancellarla, e il risultato non cambierebbe. Riesci a identificare la coppia superflua?

In questo caso non ti darò la soluzione. Per verificare se ci hai preso dovrai modificare il codice, ricompilarlo e farlo girare. Alla prossima!

Paolo Perrotta

Uguali dappertutto

Quanto è grande una variabile *int*? Cioè: quanto spazio occupa un *int* in memoria? In molti linguaggi, come ad esempio in C e in C++, la risposta cambia da un computer all'altro. Questo perché a seconda del computer su cui compiliamo, e quindi del compilatore che usiamo, una variabile intera in C può occupare più o meno spazio. Questo significa che il tipo *int* può contenere dei numeri più grandi su una certa macchina di quanto non ne contenga su un'altra. Lo stesso vale per gli altri tipi, come *long* e *double.* Nel caso delle variabili in virgola mobile, le dimensioni in memoria del tipo ne influenzano non solo la "capienza", ma anche la precisione (il numero di cifre significative).
È chiaro che in queste condizioni i programmi non possono girare nello stesso modo su tutte le macchine, nemmeno se li ricompiliamo. Ad esempio: un programma che fa calcoli matematici può usare tranquillamente un *float* per contenere una cifra su una macchina, mentre su un'altra macchina questo tipo è insufficiente a darci la precisione che vogliamo. Per risolvere questo problema, Java ha rimosso del tutto questa ambiguità. Non importa quale sia il microprocessore, la macchina virtuale Java memorizza le variabili sempre nello stesso modo. Ad esempio, un int in Java è sempre lungo quattro byte.

Geroglifici

Operatori come ||, && oppure ! potrebbero sembrarti difficili da memorizzare e distinguere. Perché scrivere "&&" anziché semplicemente *"and"*? Purtroppo Java ha preso buona parte della sua sintassi dal linguaggio C, e questo spiega perché alcuni operatori abbiano nomi così oscuri. Ma non preoccuparti. Anche se all'inizio ti sarà facile confonderti, non ci sono molte parti del linguaggio che ti richiedono di memorizzare simboli astrusi - quindi ti basterà un po' di esperienza per memorizzare perfettamente i pochi nei quali ti imbatterai.

Visual Basic .NET

☑ Immettere informazioni con caselle e pulsanti.

Controllo dell'input utente

Come impedire all'utente distratto di scrivere dati errati.

File sul CD
\soft\codice\Controlli.zip

File sul Web
www.ioprogrammo.net/files/72/controlli.zip

Click e DblClick

Il *Click* del mouse su un controllo, è una metafora molto intuitiva di un'azione che tutti eseguono frequentemente. L'evento *DblClick* si verifica quando l'utente fa doppio click sul controllo, per convenzione viene usato quando si vuole velocizzare un'operazione, facendo compiere all'utente contemporaneamente, l'azione di scelta e quella di conferma.
È importante sapere che se l'utente fa doppio click su un controllo, viene eseguito prima il codice dell'evento *Click* e poi quello dell'evento *DblClick*.

Uno dei maggiori problemi nello sviluppo di applicazioni per l'acquisizione di dati, è sempre stato quello di impedire all'utente di introdurre informazioni errate (ad esempio per errori di battitura). In questo articolo scopriamo quali sono gli accorgimenti da adottare per prevenire possibili inconvenienti. Una soluzione, laddove i possibili dati da inserire siano in numero limitato, può essere quella di sostituire il TextBox di input con un elenco di opzioni selezionabili con il mouse. In questo articolo descriveremo i controlli *CheckBox* e *RadioButton* che permettono di "presentare" opzioni di scelta ad un utente distratto, il controllo Button che permette di confermare una scelta e di compiere un'azione ed i controlli contenitore *GroupBox* e *Panel*.

METODI COMUNI

Prima di passare alla descrizione dei controlli in esame analizziamo alcuni dei metodi comuni a tutti i controlli, divisi per categoria.

Dimensioni e posizione

Quando si crea un'interfaccia utente complessa con molti controlli contenitore (oppure si utilizzano più form in un'interfaccia a documenti multipli), può essere necessario disporre tali controlli su più livelli (lo stesso dicasi per le form figlie). Per spostare e tenere traccia dei controlli è necessario modificare l'*ordine Z*. Si definisce *ordine Z* la disposizione visiva dei controlli su più livelli all'interno di una form, lungo l'asse *z* della stesso form (*profondità*). I metodi che consentono di modificare l'ordine *Z* dei controlli, sono *BringToFront* e *SendToBack*.

- **BringToFront**: permette di spostare il controllo in primo piano nell'ordine *z*. Se il controllo è figlio di un altro controllo, anche il controllo figlio viene spostato in primo piano nell'ordine *z*.
- **SendToBack** permette di spostare il controllo in secondo piano nell'ordine *z*. Se il controllo è figlio di un altro controllo, anche il controllo figlio viene spostato in secondo piano nell'ordine *z*.
- **FindForm**: permette di referenziare la form in cui si trova il controllo.

Aspetto

- **Show**: consente di rendere visibile il controllo all'utente.Mostrare il controllo equivale ad impostare la proprietà *Visible* a *true*. Dopo avere chiamato il metodo *Show*, la proprietà *Visible* restituisce un valore *true* fino a quando non viene chiamato il metodo *Hide*.
- **Hide**: consente di nascondere il controllo all'utente. Nascondere il controllo equivale ad impostare la proprietà *Visible* a *false*. Dopo avere chiamato il metodo *Hide*, la proprietà *Visible* restituisce valore *false* fino a quando non viene chiamato il metodo *Show*.
- **Invalidate**: consente di invalidare un'area specifica del controllo, determinando l'invio di un messaggio di disegno al controllo.
- **Refresh**: determina l'invalidazione dell'area client del controllo ed il nuovo disegno di quest'ultimo e degli eventuali controlli figlio.

Focus

- **Focus**: consente di impostare lo stato attivo per l'input del controllo. Alcuni controlli Windows

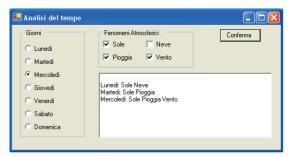



Fig. 1: Esempio di un form con più controlli.

Form non possono ricevere il focus, così come i controlli da essi derivati. Tali controlli sono: *Panel, GroupBox, PictureBox, ProgressBar, Splitter, Label* e *LinkLabel* quando nel controllo non è presente alcun collegamento.

- **GetNextControl**: consente di referenziare il controllo successivo in avanti o all'indietro nell'ordine di tabulazione dei controlli figlio.

IL CONTROLLO BUTTON

La metafora del pulsante è certamente la metafora più usata in VB, in linea di principio un pulsante è un'area dello schermo che l'utente può selezionare con un clic del mouse. Un pulsante è rappresentato, in VB.NET, dal controllo *Button*. Il controllo *Button* presenta generalmente una didascalia (ad esempio *Ok, Annulla*) ed, in particolari occasioni, un'immagine che fa comprendere immediatamente all'utente l'azione generata nel momento in cui verrà premuto il pulsante. Per modificare la descrizione del pulsante si deve usare la proprietà *Text*.

Dopo aver posizionato il pulsante sulla form si deve scrivere il codice adeguato all'azione che dovrà compiere, tale codice viene scritto solitamente nell'evento *Click*:

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                              Handles Button1.Click
    'inserire qui il codice di risposta all'evento
End Sub
```

Per inserire un'immagine nel pulsante si deve utilizzare la proprietà *Image*. È sufficiente cliccare sulla proprietà *Image*, nella finestra delle proprietà, e selezionare un'immagine dalla finestra di dialogo (è inoltre possibile usare il controllo *ImageList* che vedremo nei prossimi articoli). Per definire un pulsante come pulsante di default, si deve impostare la proprietà *AcceptButton*, della form che lo contiene, selezionando il pulsante dall'elenco a discesa. Premendo il tasto *INVIO* da un qualsiasi punto della finestra, il controllo passa al pulsante definito come pulsante di default attivandone l'evento *Click*.

Si deve prestare particolare attenzione a definire un pulsante come pulsante di default, poiché l'utente può essere portato ad usare il tasto *INVIO* per passare da un campo ad un altro, attivando il pulsante senza rendersene conto. Se il pulsante deve compiere un'operazione potenzialmente dannosa per l'applicazione (ad esempio chiudere una finestra, senza salvare i dati) è sempre auspicabile visualizzare un messaggio d'avviso per poter, eventualmente, annullare l'operazione.

Impostando la proprietà *CancelButton* della form, il pulsante selezionato verrà attivato quando viene premuto il tasto *ESC* (il tipico caso di un pulsante *Annulla*).

CHECKBOX

Il controllo *CheckBox* (casella di controllo) è rappresentato graficamente da un'etichetta con a fianco una casella. Quando il controllo viene selezionato, nella casella viene visualizzato un segno di spunta. Solitamente è utilizzato per visualizzare risposte a scelta multipla o per visualizzare una serie di opzioni tra cui selezionare quelle desiderate. Ogni volta che si clicca sul controllo, si alternano i valori *True* e *False*. Le proprietà più utilizzate sono:

- La proprietà *CheckAlign* usata per variare la posizione della casella di selezione all'interno del controllo, consente di scegliere tra nove diversi valori. La proprietà *CheckAlign*, generalmente, è utilizzata congiuntamente alla proprietà *TextAlign* permettendo di disegnare il controllo in molti modi differenti.
- La proprietà *Checked* (non è più presente la proprietà *Value*) viene usata per impostare o testare se il controllo è stato selezionato. Può essere impostata a:

  1) *False* per indicare che il controllo non è selezionato.
  2) *True* per indicare che il controllo si trova nello stato di selezionato.

L'evento generalmente più utilizzato è l'evento *Click*, che si verifica quando si modifica lo stato del controllo. Generalmente in quest'evento viene scritto il codice necessario per abilitare o disabilitare altri controlli in dipendenza della scelta effettuata. Simuliamo, ad esempio, di essere in presenza di due possibilità di scelta autoescludenti. Selezionando un CheckBox automaticamente si deve deselezionare l'altro, per questo nell'evento *Click* dei due CheckBox dovremo scrivere:

```
Private Sub CheckBox1_Click(ByVal sender As Object,
 ByVal e As System.EventArgs) Handles CheckBox1.Click
    'se Checkbox1 viene selezionato allora Checkbox2
    'deve essere deselezionato
    If CheckBox1.Checked = True Then
        CheckBox2.Checked = False
    End If
End Sub

Private Sub CheckBox2_Click(ByVal sender As Object,
 ByVal e As System.EventArgs) Handles CheckBox2.Click
    If CheckBox2.Checked = True Then
        CheckBox1.Checked = False
    End If
End Sub
```

RADIOBUTTON

Il controllo *RadioButton* (Pulsante di opzione) si comporta in modo simile al CheckBox, con la sola diffe-

Visual Basic .NET

MouseDown, MouseUp, MouseMove, MouseWheel

Nella maggior parte dei casi, gli eventi *Click* e *DblClick* saranno più che sufficienti per la gestione del mouse, ma in alcune applicazioni potrà essere necessario distinguere il click sinistro da quello destro, oppure la pressione dei tasti *Ctrl, Shift* e *Alt* contemporaneamen te al pulsante del mouse (la metafora della multiselezione), perciò si rende necessario l'uso degli eventi *MouseDown, MouseUp* e *MouseMove*. L'evento *MouseDown* viene generato quando viene premuto il pulsante del mouse (sia il sinistro sia il destro), quando il pulsante viene rilasciato viene generato l'evento *MouseUp*, e quando il mouse si sposta su un controllo viene generato l'evento *MouseMove*. Il nuovo evento *MouseWheel* viene generato quando si utilizza la rotella del mouse.

Eventi del mouse

Questi eventi ricevono le informazioni riguardanti lo stato del mouse in un oggetto *MouseEventArgs* che espone le seguenti proprietà:

• *Button* restituisce il valore (indicato sotto forma di un valore enumerato *MouseButtons*) che indica quale pulsante del mouse è stato premuto o rilasciato. Il tipo enumerato *MouseButtons* gestisce i cinque pulsanti del mouse sotto Windows 2000 e XP.

• *X* e *Y* restituiscono le coordinate *X* ed *Y* (rispetto all'angolo superiore sinistro) della posizione corrente del puntatore del mouse espresse in pixel.

• *Clicks* restituisce il numero di volte che il pulsante del mouse è stato premuto e rilasciato.

• *Delta* restituisce il numero di giri della rotellina del mouse. Un valore positivo indica che la rotella è stata ruotata in avanti, mentre un valore negativo indica che la rotella è stata ruotata all'indietro.

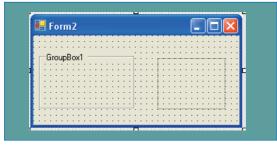



Fig. 2: **Il controllo GroupBox in azione.**

renza che in un gruppo di controlli RadioButton è possibile selezionare un solo controllo alla volta, mentre è possibile selezionare più controlli CheckBox contemporaneamente. Anche per i controlli RadioButton le proprietà più utilizzate sono le proprietà *CheckAlign* e *Checked* il cui comportamento è identico a quello delle proprietà del *CheckBox*. Solitamente i controlli *RadioButton* logicamente legati tra di loro, sono raggruppati in un controllo contenitore, tipicamente un controllo *GroupBox*. Tutti i controlli *RadioButton*, non raggruppati, presenti in una Form, appartengono allo stesso gruppo, di conseguenza sarà possibile selezionarne soltanto uno per volta (anche se, a parte il fatto di apparire sulla stessa form, non hanno nulla in comune).

MOSTRARE RADIOBUTTON E CHECKBOX COME PULSANTI

I controlli *RadioButton* ed i controlli *CheckBox* possono essere visualizzati con la veste grafica dei pulsanti per rendere l'interfaccia più accattivante. Si possono visualizzare delle icone in modo che i controlli abbiano l'aspetto grafico di un pulsante ed il comportamento dei relativi controlli: cliccando su un pulsante esso rimane nello stato di premuto (ad indicare che il pulsante è stato selezionato) fino a quando non si clicca di nuovo sul pulsante per passare allo stato di non premuto (ad indicare che il pulsante non è selezionato). Nel caso di più pulsanti *RadioButton* soltanto uno di essi resta nello stato di premuto. Si deve porre particolare attenzione a non confondere l'utente poiché i controlli *RadioButton* e *CheckBox* in modalità *pulsanti* sono uguali ai normali *Button*, si deve perciò rendere chiaro il diverso funzionamento dei controlli (per esempio raggruppandoli in un GroupBox la cui etichetta ne specifichi il funzionamento). Le proprietà da utilizzare sono:

- **Appearance**: deve essere impostata a *Button*.
- **Image**: definisce l'immagine che dovrà essere visualizzata sul *RadioButton* o sul *CheckBox*.

IL CONTROLLO GROUPBOX

Il controllo *GroupBox* permette di raggruppare controlli comportandosi come un contenitore di controlli logicamente legati fra loro, (prende il posto del controllo *Frame* presente in VB6). Generalmente il GroupBox viene utilizzato soltanto per raggruppare dei controlli, perciò non vengono gestiti i suoi eventi, e vengono solitamente modificate poche proprietà: le proprietà *Name*, *Text* e *FlatStyle*. A differenza del controllo *Frame* non è possibile disegnare un *GroupBox* senza bordi. I controlli figli devono essere creati selezionando l'icona corrispondente dalla casella degli strumenti, e disegnando il controllo all'interno dell'area del *GroupBox*.
È possibile selezionare un controllo già esistente e trascinarlo sul GroupBox. Il controllo *GroupBox* presenta alcune caratteristiche interessanti dei controlli contenitori (il principale controllo contenitore è la form):

- Se si sposta un controllo *GroupBox* vengono spostati anche i controlli che contiene;
- Se si disabilita o si rende invisibile il *GroupBox* si disabiliteranno o si renderanno invisibili tutti i controlli contenuti in esso;
- Non è possibile spostare un controllo figlio fuori dai limiti del *GroupBox*.

Nei controlli contenitore è possibile utilizzare il metodo *Scale* per spostare e ridimensionare uno o più controlli. L'effetto del metodo *Scale* è di modificare le dimensioni di un controllo in base ad un fattore di scala specificato (è possibile specificare due fattori distinti per gli assi *X* e *Y*). La caratteristica più importante di questo metodo è che la trasformazione viene applicata a tutti i controlli contenuti nel GroupBox. Se un controllo contenuto nel GroupBox è anch'esso un contenitore, le modifiche vengono applicate anche ai relativi controlli figli e così via in modo ricorsivo (pensate a cosa può succedere se si applica il metodo ad una form).

IL CONTROLLO PANEL

Il controllo *Panel* assomiglia al controllo *GroupBox*, ed è anch'esso un contenitore per altri controlli. A differenza del GroupBox non ha il bordo visibile e non ha un'etichetta (non è quindi disponibile la proprietà *Text*) mentre è in grado di gestire la proprietà *AutoScroll* e tutte le proprietà ad essa correlate (in modo che l'utente finale possa scorrerne il contenuto). È inoltre possibile ancorare o controllare la distanza dal bordo dei controlli in esso contenuto, tramite la proprietà *DockPadding*

APPLICAZIONE DI ESEMPIO

Per meglio fissare i concetti espressi finora realizziamo una semplice applicazione che mostri un prospetto riassuntivo delle condizioni meteorologiche verificatesi durante la settimana. Per il nostro scopo avremo bisogno di:

Visual Basic .NET

- 7 RadioButton che rappresentino i giorni della settimana, con i seguenti nomi: *RadioButtonLunedi, RadioButtonMartedi, RadioButtonMercoledi, RadioButtonGiovedi, RadioButtonVenerdi, RadioButtonSabato, RadioButtonDomenica;*
- 4 CheckBox che rappresentino le condizioni atmosferiche, con i seguenti nomi: *CheckBoxSole, CheckBoxPioggia, CheckBoxNeve, CheckBoxVento;*
- 2 GroupBox che raggruppino i *RadioButton* ed i *CheckBox;*
- 1 Button per confermare la condizione atmosferica verificatasi, dal nome *ButtonConferma;*
- 1 TextBox multiriga che visualizzi il prospetto riassuntivo, dal nome *TextBoxRiepilogo.*

Il codice necessario dovrà essere scritto nell'evento *Click* di *ButtonConferma*:

```
Private Sub ButtonConferma_Click(ByVal sender
        As System.Object, ByVal e As System.EventArgs)
                          Handles ButtonConferma.Click
    Dim Giorno As String
    Dim Tempo As String
    Giorno = DeterminaGiorno()
    Tempo = DeterminaTempo()
    TextBoxRiepilogo.Text = TextBoxRiepilogo.Text +
                              vbCrLf + Giorno + Tempo
End Sub
```

Sono state dichiarate due variabili, *Giorno* e *Tempo*, di tipo stringa. Alle due variabili si assegna il valore di ritorno delle due funzioni *DeterminaGiorno* e *DeterminaTempo* che si faranno carico di comporre la stringa risultato in base alle scelte dell'utente. Infine nel TextBox di riepilogo si mostrano le informazioni su ogni singola riga separandole dalla combinazione dei caratteri di ritorno a capo e avanzamento riga (*Chr(13) + Chr(10)*) rappresentati dalla costante *vbCrLf*.
La funzione *DeterminaTempo* dovrà controllare il valore della proprietà *Checked* di ogni CheckBox. Se il valore della proprietà *Checked* è pari a *True*, si dovrà comporre la stringa informativa aggiungendo la condizione atmosferica corrispondente al CheckBox se-lezionato:

```
  Private Function DeterminaTempo() As String
    Dim TmpStringa As String
 TmpStringa=""
    If CheckBoxSole.Checked = True Then
      TmpStringa = TmpStringa + " Sole"
    End If
    If CheckBoxPioggia.Checked = True Then
      TmpStringa = TmpStringa + " Pioggia"
    End If
    If CheckBoxNeve.Checked = True Then
      TmpStringa = TmpStringa + " Neve"
    End If
    If CheckBoxVento.Checked = True Then
      TmpStringa = TmpStringa + " Vento"
    End If
    DeterminaTempo = TmpStringa
  End Function
```

La funzione *DeterminaGiorno* dovrà controllare il valore della proprietà *Checked* di ogni RadioButton. Appena un RadioButton presenta un valore della proprietà *Checked* pari a *True*, si dovrà restituire il giorno corrispondente al *RadioButton* selezionato e si dovrà uscire dalla funzione (tramite l'istruzione *Exit Function*), poiché nessun altro RadioButton può avere il valore della proprietà *Checked* pari a *True*.

```
  Private Function DeterminaGiorno() As String
    If RadioButtonLunedi.Checked = True Then
      DeterminaGiorno = "Lunedi:"
      Exit Function
    End If
    If RadioButtonMartedi.Checked = True Then
      DeterminaGiorno = "Martedi:"
      Exit Function
    End If
    If RadioButtonMercoledi.Checked = True Then
      DeterminaGiorno = "Mercoledi:"
      Exit Function
    End If
    If RadioButtonGiovedi.Checked = True Then
      DeterminaGiorno = "Giovedi:"
      Exit Function
    End If
    If RadioButtonVenerdi.Checked = True Then
      DeterminaGiorno = "Venerdi:"
      Exit Function
    End If
    If RadioButtonSabato.Checked = True Then
      DeterminaGiorno = "Sabato:"
      Exit Function
    End If
    If RadioButtonDomenica.Checked = True Then
      DeterminaGiorno = "Domenica:"
      Exit Function
    End If
  End Function
```

CONCLUSIONI

Quando si dispone di un numero limitato di scelte i controlli *RadioButton* e *CheckBox* sono certamente i controlli più indicati da utilizzare, ma se le opzioni aumentano di numero non è elegante proporre all'utente una finestra piena zeppa di controlli.
VB.Net mette a disposizione altri controlli che possono essere utilizzati per proporre all'utente una serie di opzioni diverse: i controlli *ListBox* (caselle di riepilogo) e *ComboBox* (caselle combinate) che vedremo nel prossimo articolo.

Ing Luigi Buono

FindForm e parent

Per referenziare la finestra in cui si trova un controllo è bene usare la proprietà *FindForm* piuttosto che la proprietà *Parent*, poichè la proprietà *Parent* può non corrispondere al controllo *Form* restituito dal metodo *FindForm*. Se, ad esempio, un controllo *RadioButton* è contenuto in un controllo *GroupBox* ed il controllo *Group Box* si trova in una Form, l'oggetto *Parent* del controllo *RadioButton* è *GroupBox* mentre soltanto l'oggetto *Parent* del controllo *GroupBox* sarà la *Form.*

C#

✓ Approfondimento sul concetto di ereditarietà.

Approfondimento sul concetto di ereditarietà

Continua il nostro viaggio nei meandri dell'ereditarietà. In questa lezione si parlerà di costruttori, di nascondimento e di redifinizione. Si scopriranno, inoltre, alcuni nuovi pregi dell'ereditarietà.

File sul CD
\soft\codice
\csharp17.zip

File sul Web
www.ioprogrammo.net/files/72/csharp17.zip

La precedente lezione ha introdotto il fondamentale concetto di *ereditarietà*. Abbiamo visto cosa significa creare una gerarchia di classi in rapporti di ereditarietà, attraverso alcune applicazioni pratiche. Sono stati svelati i primi misteri collegati all'argomento. Questa lezione approfondisce la precedente, elencando alcune norme ed alcune possibilità che è indispensabile conoscere per fare un uso proficuo dell'ereditarietà.

EREDITARIETÀ E COSTRUTTORI

Finora abbiamo parlato di ereditarietà, ma non abbiamo ancora preso in considerazione i rapporti che corrono tra i costruttori di una superclasse e quelli delle sue sottoclassi. **Prima regola:** *i costruttori non vengono ereditati.* Se una classe *X* dispone di un costruttore con tre argomenti di tipo *string*, giusto per citare un esempio, una sua sottoclasse *Y* non disporrà automaticamente dello stesso costruttore. Vale sempre la medesima norma, anche nei casi di ereditarietà: se nessun costruttore è esplicitamente dichiarato, la classe verrà automaticamente dotata di un costruttore di default, privo di argomenti, che non compie alcuna operazione. Al contrario, se si definisce anche un solo costruttore, quello di default privo di argomenti non sarà più fornito automaticamente dal linguaggio. **Seconda regola:** quando si istanzia una sottoclasse, *almeno uno dei costruttori della sua superclasse deve essere richiamato implicitamente o esplicitamente*. Andiamo per casi. Quando si istanzia una sottoclasse, mediante uno qualsiasi dei suoi costruttori, viene automaticamente richiamato il costruttore senza argomenti della sua superclasse. La verifica è semplice:

```
class A {
  public A() {
    System.Console.WriteLine("Costruttore di A");}
}
class B : A {
  public B() {
    System.Console.WriteLine("Costruttore di B"); }
}
```

Eseguiamo un test servendoci delle due classi appena definite:

```
class Test {
  public static void Main() { B b = new B(); } }
```

L'output prodotto è il seguente:

Costruttore di A
Costruttore di B

Come è possibile osservare, un oggetto di tipo *B* è stato costruito eseguendo prima il costruttore della superclasse *A* e poi quello della sottoclasse *B*, in questo preciso ordine. La norma vale finché la superclasse dispone di un costruttore privo di argomenti. In caso contrario, è necessario specificare esplicitamente quale dei costruttori della superclasse vada richiamato, all'interno di ogni costruttore della sottoclasse realizzata. A tal fine, ilc ostruttore della classe derivata va dichiarato al seguente modo, facendo uso della parola *base*:

```
costruttore-derivato(argomenti) : base(argomenti) {
  // ... }
```

Ecco un esempio:

```
class A {
  public int x;
  public A(int x) { this.x = x; }
  public int getX() { return x; }
}
class B : A {
```

Casting fra classi in rapporti di ereditarietà

Si supponga di avere due classi A e B, nelle quali B deriva da A. Sappiamo che è possibile fare quanto segue, cioè memorizzare un'istanza di *B* in un riferimento di tipo *A*:

```
A rifTipoA = new B();
```

In questo modo, sull'oggetto creato possono essere richiamati i membri tipici dell'interfaccia di A. Se, ad un certo punto, si desidera fare marcia indietro e tornare ad un riferimento di tipo *B*, è possibile impiegare il casting:

```
B rifTipoB = (B)rifTipoA;
```

```
  public int y;
  public B(int x, int y) : base(x) { this.y = y; }
  public int getY() { return y; }
}
class Test {
  public static void Main() {
    B b = new B(5, 7);
    System.Console.WriteLine("b.x = " + b.getX());
    System.Console.WriteLine("b.y = " + b.getY()); }
}
```

In *A* è stato definito un solo costruttore, che accetta un parametro di tipo *int*. In questo modo, nessun costruttore privo di argomenti è più disponibile. In *B* è stato introdotto un costruttore con due argomenti di tipo *int*. Il primo di essi, *x*, viene fornito al costruttore della superclasse *A*, attraverso una chiamata a *base*:

```
public B(int x, int y) : base(x)
```

Eseguendo il programma, si ottiene il seguente output, che rispetta le aspettative:

```
b.x = 5
b.y = 7
```

Concludiamo il paragrafo con un ulteriore esempio, ottenuto revisionando le classi *Persona* e *Studente* esaminate nel corso dell'appuntamento precedente:

```
class Persona {
  public string nome;
  public string cognome;
  public Persona(string nome, string cognome) {
    this.nome = nome;
    this.cognome = cognome; }
  public string getNome() { return nome; }
  public string getCognome() { return cognome; }
}
```

In questa versione di *Persona*, sono stati rimossi i metodi *setNome()* e *setCognome()*. Ora è possibile impostare i campi *nome* e *cognome* direttamente alla creazione di un oggetto *Persona*, servendosi del costruttore della classe. Fin qui, nessun problema. Ma cosa accade nella sottoclasse *Studente*? Torniamo a considerarla nella sua formulazione originaria:

```
class Studente : Persona {
  public string matricola;
  public void setMatricola(string matricola) {
    this.matricola = matricola;}
  public string getMatricola() { return matricola; }
}
```

Mettiamo insieme le due classi in un *Main()* d'esempio:

```
class Test {
  public static void Main() {
    Studente s = new Studente();
    s.setMatricola("09204167");
    System.Console.WriteLine("Nome: " + s.getNome());
    System.Console.WriteLine("Cognome: "
                                        + s.getCognome());
    System.Console.WriteLine("Matricola: " + s.getMatricola()); }
}
```

La compilazione non riesce. L'errore è il seguente:

error CS1501: Nessun overload del metodo "Persona" accetta "0" argomenti

La classe *Studente* non ha costruttore. Pertanto, il compilatore gliene fornisce uno di default, privo di argomenti, che non compie alcuna operazione. Nella sua dichiarazione, quindi, non ci sarà nessuna chiamata a *base*. Ne consegue che, alla creazione di un oggetto *Studente*, sarà richiesto un costruttore di *Persona* privo di argomenti. Giacché non esiste un costruttore di questo tipo, l'operazione è impossibile. Per questo, il compilatore interrompe il proprio operato, impedendo anzitempo la catastrofe. Quindi, o dotiamo *Persona* di un costruttore privo di argomenti, oppure definiamo uno o più costruttori in *Studente* che facciano uso di *base*. Optiamo per la seconda soluzione, che tecnicamente è quella più corretta:

```
class Studente : Persona {
  public string matricola;
  public Studente(string nome, string cognome, string
                matricola) : base(nome, cognome) {
    this.matricola = matricola; }
  public string getMatricola() { return matricola; }
}
```

Missione compiuta! Da questo momento in poi, potremo creare istanze di *Studente* ricalcando il seguente modello:

```
Studente s = new Studente("Mario", "Rossi", "09204167");
```

Aggiustiamo il *Main()* in tal direzione:

```
class Test {
  public static void Main() {
    Studente s = new Studente("Mario", "Rossi",
                                        "09204167");
    System.Console.WriteLine("Nome: " + s.getNome());
    System.Console.WriteLine("Cognome: "
                                        + s.getCognome());
    System.Console.WriteLine("Matricola: " + s.getMatricola()); }
}
```

Ora le tre classi *Persona*, *Studente* e *Test*, insieme, possono essere compilate ed eseguite. L'output è:

Nome: Mario
Cognome: Rossi

C#

Classe Studente

La classe *Studente* non ha costruttore. Pertanto, il compilatore gliene fornisce uno di default, privo di argomenti, che non compie alcuna operazione. Nella sua dichiarazione, quindi, non ci sarà nessuna chiamata a *base*.

C#

Matricola: 09204167

NASCONDIMENTO DEI NOMI

I membri delle classi derivate possono avere lo stesso nome dei membri delle corrispondenti classi base. In una situazione di questo tipo, l'elemento della classe derivata nasconderà quello della sua classe base:

```
class A { public int i = 1; }
class B : A { public int i = 2;}
class Test {
  public static void Main() {
    B b = new B();
    System.Console.WriteLine(b.i); }
}
```

La proprietà i definita nella classe derivata *B* nasconde l'omonima proprietà della classe base *A*. Nonostante la compilazione riesca senza problemi, il compilatore emette un segnale di avviso *(warning):*

warning CS0108: La parola chiave new è necessaria in "B.i" perché nasconde il membro ereditato "A.i".

In sostanza, il compilatore non sa se il nascondimento di *i* sia intenzionale o meno, per questo ci avvisa che nella classe base *A* è già presente una proprietà chiamata *i*, che stiamo nascondendo. Se si vuole evitare questo warning, è sufficiente far capire al compilatore che il nascondimento è del tutto intenzionale. Per riuscire nell'intento, è possibile fare un particolare uso della parola chiave *new*:

```
class B : A { public new int i = 2;}
```

In questo modo, il programmatore dichiara la consapevolezza del nascondimento, ed il compilatore non avrà più nulla da segnalare. Si è visto un esempio basato su una proprietà, ma il medesimo discorso vale anche nei confronti dei metodi:

```
class A {
  public void saluta() {
    System.Console.WriteLine("A ti saluta!");}
}
class B : A {
  public new void saluta() {
    System.Console.WriteLine("B ti saluta!"); }
}
class Test {
  public static void Main() {
    B b = new B();
    b.saluta(); }
}
```

Eseguendo il programma, come oramai dovrebbe essere facile intuire, l'output sarà: *B ti saluta!*
Se, ad un certo punto, si desidera fare in modo che la classe derivata richiami ed utilizzi un membro nascosto della sua classe base, si può fare un uso particolare della parola chiave *base*:

```
class B : A {
  public new void saluta() {
    base.saluta();
    System.Console.WriteLine("B ti saluta!"); }
}
```

La chiamata a *base.saluta()* fa in modo che venga eseguito il metodo *saluta()* definito nella superclasse A. L'output ora sarà:

A ti saluta!
B ti saluta!

UTILIZZO DELLE CLASSI DERIVATE

Un aspetto che rende utile la pratica dell'ereditarietà consiste nel fatto che una variabile che fa riferimento ad una superclasse può fare riferimento anche ad una sua qualsiasi sottoclasse. Detto a parole sembra complicato, anche se in realtà non lo è. Un codice esemplificativo ci sarà d'aiuto:

```
class A {
  public void prova() {
    System.Console.WriteLine("prova() di A");}
}
class B : A {
  public new void prova() {
    System.Console.WriteLine("prova() di B"); }
}
class C : B {
  public new void prova() {
    System.Console.WriteLine("prova() di C");
  }
}
class Test {
  public static void Main() {
    A a1 = new A();
    A a2 = new B();
    A a3 = new C();
    a1.prova();
    a2.prova();
    a3.prova();
  }
}
```

Ci sono tre classi, chiamate *A*, *B* e *C*, disposte secondo la seguente gerarchia:

```
object
 |
 +- A
   |
   +- B
```

Membri classi derivate

I membri delle classi derivate possono avere lo stesso nome dei membri delle corrispondenti classi base. In una situazione di questo tipo, l'elemento della classe derivata nasconderà quello della sua classe base.

```
       |
       +- C
```

Nella classe di verifica *Test*, si eseguono le seguenti dichiarazioni:

```
A a1 = new A();
A a2 = new B();
A a3 = new C();
```

Sono state dichiarate tre variabili, tutte di tipo *A*. In *a1*, come consuetudine, è stato memorizzato il riferimento ad un oggetto di tipo *A*. Fin qui, nulla di nuovo. In *a2* e *a3*, invece, abbiamo memorizzato delle istanze di *B* e di *C*. Ciò è possibile grazie alla norma illustrata in apertura di paragrafo. Successivamente, abbiamo invocato il metodo *prova()* su ogni istanza creata, ottenendo l'output:

```
prova() di A
prova() di A
prova() di A
```

Nonostante il nascondimento del metodo *prova()* effettuato in *B* e *C*, è stato richiamato tre volte il metodo *prova()* definito in *A*. Questo avviene poiché le istanze di *B* e *C* sono state memorizzate in un riferimento di tipo *A*. Dunque, la natura del riferimento alla classe rende più o meno valido il nascondimento di un membro. Se l'istanza di *C* venisse memorizzata in un riferimento di tipo *B*, si otterrebbe una chiamata a *prova()* di *B*. Si può dimostrare:

```
class Test {
  public static void Main() {
    B b = new C();
    b.prova();
  }
}
```

L'output sarà:

prova() di B

RIDEFINIZIONE DEI METODI

La ridefinizione di un metodo è qualcosa di apparentemente simile al nascondimento dello stesso. La sostanza, infatti, è la stessa: si dota la classe derivata di un metodo con la stessa firma di un altro metodo presente nella classe base. La ridefinizione, però, entra in gioco quando tale metodo della classe base è definito virtuale, con la parola chiave *virtual*:

```
class A {
  public virtual void prova() {
    System.Console.WriteLine("prova() di A"); }
}
```

Il metodo *prova()* di *A*, ora, è virtuale. Può, quindi, essere sottoposto a ridefinizione. La ridefinizione di *prova()*, in una classe derivata da *A*, è simile al nascondimento, con l'eccezione che va usata la parola chiave *override* al posto della già nota *new:*

```
class B : A {
  public override void prova() {
    System.Console.WriteLine("prova() di B");
  }
}
```

Eseguiamo un test:

```
class Test {
  public static void Main() {
    A a = new A();
    B b = new B();
    a.prova();
    b.prova(); }
}
```

L'output è:

prova() di A
prova() di B

Fin qui, nessuna novità riscontrata nei confronti del nascondimento. Proviamo, però, a memorizzare l'istanza di *B* in un riferimento di tipo *A*:

```
class Test {
  public static void Main() {
    A a1 = new A();
    A a2 = new B();
    a1.prova();
    a2.prova(); }
}
```

L'output non cambia:

prova() di A
prova() di B

Quindi, riassumendo, la ridefinizione fa in modo che la selezione del metodo da richiamare dipenda dalla natura stessa dell'oggetto, e non dal tipo del riferimento in cui lo si memorizza, come avveniva invece con le tecniche di nascondimento. Anche se l'istanza di *B* è stata memorizzata in un riferimento di tipo *A*, la versione di *prova()* richiamata è quella di *B*.

CONCLUSIONI

Termina lo spazio a disposizione per questo mese, ma l'argomento ereditarietà non è ancora esaurito. Vi aspetto dunque per l'appuntamento successivo, quando il discorso sarà ripreso da dove lo si interrompe ora.

Carlo Pelliccia

C#

Bibliografia

• GUIDA A C#
Herbert Schildt
(McGraw-Hill)
ISBN 88-386-4264-8
2002

• INTRODUZIONE A C#
Eric Gunnerson
(Mondadori Informatica)
ISBN 88-8331-185-X
2001

• C# GUIDA PER LO SVILUPPATORE
Simon Robinson e altri
Hoepli, 2001
ISBN 88-203-2962-X

C++

Un punto cardine del linguaggio: le librerie.

Namespace e stringhe

A partire da questa puntata daremo un'occhiata alle principali caratteristiche delle librerie standard del C++. Iniziamo il nostro viaggio dalla libreria per il trattamento delle stringhe. Ben presto abbandoneremo i char*!

File sul CD
\soft\codice\
Lez_15.zip

File sul WEB
www.ioprogrammo.net
/files/72/Lez_15.zip

Come lo stesso Stroustrup asserisce, nessun programma è scritto partendo dalle istruzioni basilari del linguaggio di programmazione: prima si sviluppano le librerie contenenti le funzioni di cui si fa uso; assumendo queste librerie come base di partenza, quindi, si procede allo sviluppo di ulteriori programmi.
Questo genere di approccio è quello seguito nella definizione della *libreria standard per il C++*: dopo la definizione del linguaggio, si è proceduto alla definizione di librerie di supporto che rendessero possibile programmare in C++ senza dover "reinventare la ruota". Ovviamente, non è obbligatorio (anche se fortemente consigliato) fare uso di tali librerie: i creatori del C++ hanno lasciato la massima libertà ai programmatori, i quali possono benissimo ignorare l'esistenza della libreria standard (un po' come abbiamo fatto noi fino ad ora). Essenzialmente, la libreria standard del C++ include tutte quelle utilità la cui funzione possa essere necessaria all'interno di un qualsiasi programma, sia esso avanzato o basilare. Essa comprende le principali strutture dati e i principali algoritmi per lavorare con tali strutture. Tutte le utilità introdotte sono state inserite seguendo i criteri di frequenza di *utilizzo* (la libreria include solo quelle cose che vengono usate praticamente da tutti i possibili programmatori C++), *usabilità* (facilità di utilizzo e apprendimento, nei limiti del possibile) e *ottimizzazione* (l'uso delle funzionalità fornite dalla libreria non deve provocare considerevoli *overhead*, cioè "appesantimenti", nei programmi). All'interno della libreria standard si trovano essenzialmente le seguenti cose:

- la libreria standard del C;
- supporto per stringhe e manipolatori di stream;
- supporto per diversi tipi di "*oggetti contenitore*", cioè le principali strutture dati utilizzabili nei programmi (es. pile, code, vettori);
- supporto per il calcolo numerico e scientifico (es. numeri complessi, operatori logici);
- supporto per l'ambiente run-time (es. gestione delle *eccezioni*).

Spesso, si sente parlare della *Standard Template Library* (STL): essa non è la libreria standard del C++, ma un suo sottoinsieme, scritto in gran parte da Alexander Stepanov. Ci si riferisce alla STL solitamente per indicare l'insieme dei *contenitori, iteratori* ed *algoritmi* inclusi nella libreria standard.

I NAMESPACE

Molte puntate fa (nella prima puntata, in realtà) parlammo dello *scope*, dicendo che esso rappresenta l'*ambito di visibilità* di una variabile. Il concetto di *namespace* è molto vicino al concetto di *scope*. Si può pensare ai *namespace* come a degli "spazi dei nomi": definire un *namespace* significa raggruppare sotto un nome comune un insieme di elementi correlati, mantenendo quindi separati gli elementi che appartengono a namespace diversi. I *namespace* sono molto utili anche, ad esempio, nella creazione di librerie di funzioni, oppure per separare versioni diverse di codici o classi.
Usare un *namespace* significa rendere visibili dei nuovi elementi all'interno dei nostri programmi. Vediamo un esempio di definizione di un *namespace*:

```
namespace Acquario
{
  class Pesce
  {
    //codice della classe
```

```
  };
}
```

Si noti che la definizione di un *namespace* non termina con il ";", a differenza di ciò che si fa per le classi. Con la definizione precedente, si è inteso dire che solamente usando il *namespace Acquario*, la classe *Pesce* sarà visibile al nostro programma. L'utilità dei *namespace* diventa evidente nel momento in cui si ha a che fare con *classi omonime*. Se si avesse la seguente definizione di *namespace*:

```
namespace LaghettoArtificiale
{
  class Pesce
  {
    //codice della classe
  };
}
```

le classi definite nel nuovo *namespace* (e analogamente per tutti gli altri elementi nel *namespace*) sarebbero altre classi, omonime a quelle definite nel *namespace Acquario*, ma appartenenti ad un *namespace* diverso. Le classi così definite sono utilizzabili anche in contemporanea, mediante un uso opportuno dell'operatore di *scope resolution "::"*.
Ad esempio:

```
LaghettoArtificiale::Pesce p1("Trota");
```

e

```
Acquario::Pesce p2("Discus");
```

sarebbero due oggetti di due classi diverse (anche se omonime), e quindi potrebbero coesistere all'interno dello stesso programma: infatti, la classe *Pesce* di cui fanno parte è nel primo caso quella del *namespace LaghettoArtificiale*, nel secondo quella del *namespace Acquario*. La definizione di un *namespace* può essere anche divisa in più file, utilizzando le direttive al compilatore *#ifndef e #define*. Questo è molto utile per estendere un namespace già esistente, aggiungendovi, ad esempio, altre classi. La struttura del codice per tale estensione è la seguente:

```
//file acquario1.h
#ifndef ACQ1
#define ACQ1
namespace Acquario
{
//...contenuto del namespace...
}
#endif
//file acquario2.h
#ifndef ACQ2
#define ACQ2
#include "acquario1.h"
namespace Acquario
{
//...estensione del namespace Acquario...
}
#endif
```

COSA SONO LE STRINGHE?

Vedremo molto presto che il discorso appena fatto sui *namespace* ci tornerà utile per l'utilizzo delle librerie standard che implementano le stringhe. Ma cosa si intende per "stringa"? Per un programmatore C, una stringa altro non è che un *array di caratteri* che termina con un carattere speciale detto *"terminatore di stringa"* (rappresentato da *"\0"*, cioè uno zero binario, al termine della stringa). Tale carattere è necessario per potere utilizzare le funzioni standard del C per la manipolazione di stringhe. Una rappresentazione così "grezza" del concetto di stringa, presenta però notevoli problemi, ad esempio la necessità di ridimensionare a mano la dimensione dell'array qualora si rendesse necessario ingrandire la stringa, oppure l'attenzione da parte del programmatore a non scrivere su zone di memoria ormai deallocate. È per questo che si è deciso di dotare il C++ di una libreria standard per la manipolazione delle stringhe attraverso un nuovo tipo: *string*. In C++ un oggetto di tipo *string* (che da ora in poi sarà chiamato, per semplicità, "stringa") ha due vantaggi rispetto al suo analogo in C:

- è un tipo vero e proprio, con tanto di funzioni che ne fanno uso (ad es. l'I/O con i classici operatori è già ridefinito per fare uso anche di oggetti di tipo stringa), e contiene al proprio interno molte informazioni utili per il compilatore ed il programmatore (ad es. la lunghezza della stringa, la sua locazione nella memoria, il numero di caratteri da cui è costituita);
- una stringa non è un array di caratteri che termina con un "carattere di terminazione": nessuna delle funzioni della libreria standard fa implicito riferimento all'esistenza di tale carattere né, quindi, il programmatore se ne deve curare.

Piccolo dazio da pagare a cotanta potenza espressiva è il dovere includere l'intestazione <*string*> nel proprio codice, oltre ad aggiungere la seguente riga di codice:

```
using namespace std;
```

che indica al compilatore che si farà riferimento in seguito al *namespace std* (ecco il perché del paragrafo precedente!). *"std"* sta per *"standard"* ed è indicativo del fatto che stiamo utilizzando proprio le librerie con questa caratteristica.
Bene, ora che abbiamo visto come informare il compilatore della nostra volontà di utilizzare il tipo *string*, vediamo come utilizzarlo.

Contatta gli autori!

Se hai suggerimenti, critiche, dubbi o perplessità sugli argomenti trattati e vuoi proporle agli autori puoi scrivere agli indirizzi:

alfredo.marroccelli@libero.it **(Alfredo)**
e
marcodelgobbo@libero.it **(Marco)**

Questo contribuirà sicuramente a migliorare il lavoro di stesura delle prossime puntate.

USO DELLE STRINGHE

Abbiamo detto che possiamo adoperare *string* come un tipo di variabile predefinita a tutti gli effetti. È possibile dichiarare una variabile di tipo *string* nel seguente modo (ormai conosciuto):

```
string s1;
```

questa stringa non contiene alcun carattere e la sua lunghezza è 0. Tuttavia quello che molto spesso si fa è inizializzare una stringa contestualmente alla sua dichiarazione, cioè assegnare alla stringa un contenuto all'interno della stessa istruzione con la quale si dice al compilatore che la stringa esiste. Ci sono molti modi per fare questo e sicuramente una panoramica con l'uso di qualche riga di codice risulterà più espressiva di mille parole:

```
//inizializzazione con costruttore a 1 argomento
string s2("Sono la stringa numero 2!");
//inizializzazione da costante
string s3 = "Io la numero 3!";
//s3bis e'  inizializzata con il contenuto di s3
string s3bis(s3);
```

nulla di sconvolgente quindi (almeno si spera!). Questi che abbiamo elencato sono i metodi più semplici e in realtà ve ne sono altri che consentono (ai programmatori più smaliziati) una notevole flessibilità nell'uso delle stringhe. Tuttavia tutti questi ulteriori modi possono essere ricondotti ai quattro sopra elencati, per cui sarebbe opportuno che il bravo programmatore C++ conoscesse questi ultimi a menadito.
Tutto questo è interessante ma a dire il vero non si discosta poi molto dall'utilizzo *"raw"* di stringhe, intese come semplici array di caratteri. Come abbiamo già avuto modo di dire, quello che rende veramente utili le librerie standard, sono il gran numero di funzioni accessorie già pronte all'utilizzo che esse mettono a disposizione.
Tra le prime funzioni di questo tipo che vogliamo menzionare, ci sono indubbiamente gli *overload degli operatori* più comuni. Utilizzando il tipo *string*, infatti, è possibile ad esempio *concatenare* due stringe semplicemente utilizzando l'operatore "+", nel modo più intuitivo che ci si possa aspettare:

```
string s1 = "La mia prima";
string s2 = " stringa concatenata";
string s1_s2 = s1 + s2; //"La mia prima stringa
                                        concatenata" :-)
```

in questo caso la concatenazione è avvenuta contestualmente a una inizializzazione, ma è del tutto possibile concatenare una stringa a un'altra stringa già esistente, ad esempio:

```
string s3 = "Seconda";
s3 = s3 + s2; //"Seconda stringa concatenata"
```

Fortunatamente i progettisti delle librerie standard non ci fanno mancare nulla, per cui è anche possibile scrivere il precedente frammento di codice condensandolo nella più breve istruzione:

```
s3 += s2; //alternativo al precedente
```

che utilizza l'operatore "+=" e ha il medesimo significato. Dopo avere concatenato tutte queste stringhe, ovviamente la cosa più intuitiva che ci viene in mente di fare è... STAMPARE una stringa a schermo! Nulla di più semplice: basta utilizzare l'overload dell'operatore di estrazione "<<" in concomitanza col consueto stream standard di output a consolle (per gli amici "*cout*"); il codice è il seguente:

```
string s4 = "Prova di stampa con cout";
cout << s4;
```

e, una volta eseguito, stamperà:

Prova di stampa con cout

come è lecito attendersi. Ovviamente è possibile continuare ad usare la normale *concatenazione di flussi* per *cout*, ma, si badi bene, questo non ha nulla a che vedere con la *concatenazione di stringhe*, sebbene in molti casi il risultato possa essere il medesimo:

```
string s5 = "Ci";
string s6 = "ao!";
//stampo una stringa concatenata
cout << (s5 + s6) << endl;
//concateno i FLUSSI di stampa!
cout << s5 << s6 << endl;
```

Questo codice stamperà:

Ciao!
Ciao!

anche se i due saluti sono ottenuti in maniera del tutto diversa. Molto comodo risulta, al contrario di ciò che accade utilizzando gli array di caratteri, l'overload dell'operatore di inserimento ">>", che consente di immettere da tastiera una stringa di lunghezza arbitraria senza stare a preoccuparci di dimensionare in maniera corretta l'array (questo è comunque un discorso valido in ogni caso). Per fare un esempio di utilizzo di questo operatore, rispolveriamo un programmino che i lettori più affezionati sicuramente ricorderanno, e modifichiamolo utilizzando le stringhe:

```
#include <iostream>
#include <string>
using namespace std;
void main()
{
   cout << "Ciao! Come ti chiami? ";
```

Stringhe

Possiamo adoperare *string* come un tipo di variabile predefinita a tutti gli effetti. È possibile dichiarare una variabile di tipo *string* nel seguente modo (ormai conosciuto):

string s1

```
string nome;
   cin >> nome;
    cout << "Ciao " << nome << "! Io sono HAL 9000..."; }
```

Dovreste già sapere cosa produce in output questo programma :-) Una cosa importante da notare è che in questo caso *cin* utilizza come *terminatore di stringa* il carattere *SPAZIO* per cui si potrebbe produrre qualche risultato indesiderato:

Ciao! Come ti chiami? Homer Simpson
Ciao Homer! Io sono HAL 9000...

anziché:

Ciao! Come ti chiami? Homer Simpson
Ciao Homer Simpson! Io sono HAL 9000...

Per ottenere il secondo comportamento può essere utile utilizzare la funzione *getline():* si sostituisce:

```
cin >> nome;
```

con

```
getline(cin,nome);
```

La funzione *getline()* inserisce nella stringa l'intera sequenza di caratteri battuta da tastiera prima di premere invio.

FACCIAMO PURE PARAGONI

Un altro overload di indubbia utilità nella programmazione pratica è senz'atro l'operatore di uguaglianza "==". Utilizzandolo, infatti, è possibile testare in maniera facile e elegante dal punto di vista sintattico, una condizione di uguaglianza tra due stringhe. Ecco un esempio:

```
void test(const string& a, const string& b) {
   cout << "Le stringhe (" << a << ") e (" << b << ") ";
if (a == b)
   cout << "SONO uguali!\n";
else
    cout << "NON sono uguali!\n";}
//... all'interno del codice...
string s7 = "Ciao!";
string s8 = "Ciao!";
string s9 = "Hello!";
test(s7,s8);
test(s7,s9);
```

Questo codice produrrà in output:

Le stringhe (Ciao!) e (Ciao!) SONO uguali!
Le stringhe (Ciao!) e (Hello!) NON sono uguali!

Ovviamente è possibile utilizzare anche l'overload dell'operatore di disuguaglianza "!=" che ha il significato opposto. Le librerie standard ridefiniscono in realtà anche molti altri operatori del C++, tra i quali gli operatori di maggioranza/minoranza (">","<",">=","<=").
Il significato di *"stringa maggiore di un'altra stringa"* potrebbe apparire leggermente più ostico di quanto visto sino ad ora, ma in realtà il concetto è abbastanza semplice. La comparazione che svolgono questi operatori è di tipo *lessicale*. Questo vuol dire che *una stringa sarà valutata maggiore di un'altra stringa se il primo carattere (partendo da sinistra) per il quale le due stringhe differiscono viene, in ordine alfabetico, dopo il corrispondente carattere nella seconda stringa*. Un po' contorto ma molto semplice. Ad esempio avendo le seguenti stringhe:

```
string casa = "casa";
string castello = "castello";
```

possiamo vedere che il primo carattere per cui esse differiscono è quello in posizione 3 (cominciando a contare da 0). Per la variabile *casa* questo carattere è *"a"*, mentre per *castello* questo carattere è *"t"*, questo vuol dire che la condizione:

```
(castello > casa)
```

restituirà il valore *true* (vero), proprio perché la *"t"* viene dopo la *"a"* nell'alfabeto. È importante notare che è definito un ordinamento per tutti i caratteri utilizzati da un calcolatore (uno di questi ordinamenti è ad esempio il set ASCII), per cui anche se le stringhe che adoperiamo contengono caratteri non contenuti nell'alfabeto (ad esempio *"!" "?" "&"* ecc.) sarà sempre possibile effettuare una comparazione, in quanto per essa viene utilizzato proprio il numero intero associato al singolo carattere della stringa. Per i caratteri dell'alfabeto vale l'ordine alfabetico semplicemente perché i numeri interi associati alle lettere seguono lo stesso ordine che segue l'alfabeto (e cioè "a" viene prima di "b" che viene prima di "c" e così via...).

CONCLUSIONI

In questa lezione abbiamo dato una panoramica molto generale di quelle che sono le stringhe nelle librerie standard del C++. L'utilizzo delle stringhe è una delle cose più comuni che si possa ritrovare a fare un programmatore quando scrive codice e per questo sapere padroneggiare, con una certa sicurezza, un set di librerie universalmente presenti, e per di più notevolmente efficienti, è senza dubbio molto importante. Quello che abbiamo appena visto non è altro che una goccia nel mare delle librerie standard e solo una piccola parte di tutte le funzionalità messe a disposizione del tipo *string*.
Vedremo nella prossima lezione di dare ulteriori nozioni sull'utilizzo di questa importante caratteristica. Non mancate!

Alfredo Marroccelli e Marco Del Gobbo

Sul Web

A chi volesse approfondire la sua conoscenza sulle librerie standard del C++, consigliamo il validissimo libro "Thinking in C++ - 2nd ed. – Volume 2" di Bruce Eckel e Chuck Allison, che rappresenta sicuramente un ottimo riferimento per i programmatori più avanzati (o aspiranti tali) ed è oltretutto disponibile gratuitamente per il download, partendo dall'indirizzo

http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html

Se volete invece dare un'occhiata a una reference delle funzioni standard del C per la manipolazione di stringhe (o meglio: di array di caratteri terminati da "\0" :-) consultate l'indirizzo:

http://www.cplusplus.com/ref/cstring/

MATLAB

La simulazione e modellazione di fenomeni fisici.

Applicazione di modelli matematici

Stephen Hawkings nel suo libro "Dal big bang ai buchi neri" ci fornisce una suggestiva definizione di modello matematico: "un insieme di regole che mettono in relazione le quantità presenti nel modello con le osservazioni che facciamo della realtà. Il modello esiste solo nella nostra mente e non possiede alcuna realtà".

File sul CD
\soft\codice
\matlab72.zip

File sul Web
www.ioprogrammo.net
/files/72/matlab72.zip

Un modello, per essere un buon modello, deve soddisfare due requisiti: descrivere con precisione (accuratezza) una grande classe di osservazioni sulla base di un modello contenente solo qualche elemento arbitrario e fare predizioni ben definite sui risultati di future osservazioni. Per esempio la legge di gravitazione di Newton." Quando descriviamo il comportamento di un fenomeno fisico dobbiamo ricorrere al concetto di modello matematico. Il modello é sempre e comunque virtuale e il software non é altro che una rappresentazione stabile, riproducibile e riutilizzabile di un pensiero. Esso é costantemente sotto inchiesta e suscettibile di confutazioni o, nel migliore dei casi, di miglioramenti. Lo scopo ultimo di un modello é proprio quello migliorarsi nel tempo sino a divenire adatto a descrivere una realtà fisica con un grado di approssimazione sufficiente per gli usi per cui é nato. Il miglioramento di un modello passa attraverso due filoni principali di attività del modellatore: la determinazione della struttura del modello e l'identificazione dei suoi parametri (le costanti ignote). Questi due aspetti ci offrono due differenti problemi da risolvere: il primo ci pone nella situazione di dover determinare quali leggi descrivano appropriatamente i fenomeni e con quale grado di accuratezza. Una volta compiuto questo passo dovremo tuffarci all'interno del modello per determinare quei parametri ancora sconosciuti che lo renderanno adatto a descrivere appropriatamente la realtà dei fatti. Eventualmente, possiamo reiterare il processo fino ad arrivare ad uno stadio di sviluppo che sia soddisfacente. Qui il modello ci viene in soccorso poiché possiamo mettere alla prova le nostre idee prima di sperimentarle praticamente. É questo un passo importantissimo del processo di indagine poiché é ora possibile scartare le idee peggiori, suffragare le migliori e lavorare creativamente per farne nascere di nuove. E ognuna deve prendere la via delle prime: essere verificata per mezzo del modello, passare il setaccio dei test e qualificarsi per essere sperimentata nella pratica. Anche se può sembrare un processo estremamente semplice e intuitivo si tratta, invece, di una parte strutturalmente importante del metodo scientifico: una volta identificato un problema é necessario fare delle ipotesi sulle sue cause, queste vanno obbligatoriamente verificate con una batteria di test che possa, in maniera inequivocabile, rigettarle o confermarle. L'analisi dei risultati attesi ci dice quali ipotesi siano state confutate e quali altre confermate. La scienza e la tecnologia progrediscono in questa maniera da molti secoli. Quando ci troviamo di fronte ad un fenomeno noi dobbiamo comportarci nella stessa maniera se vogliamo che i risultati dei nostri studi e dei nostri sforzi siano riconosciuti e considerati autorevoli. Un modello matematico altro non é che un prototipo della realtà e gli strumenti software oggi disponibili ci permettono di implementare modelli molto complessi consentendoci di scoprire l'intima natura dei fenomeni che stiamo esaminando. Come tale, ci consente di esplorare idee che sarebbero costose da esaminare per mezzo di esperimenti reali, di effettuare virtualmente test che nella realtà sono distruttivi, di studiare fenomeni non riproducibili in laboratorio, di osservare l'evoluzione nel tempo di fenomeni troppo veloci o troppo lenti, di prevedere comportamenti difficili da immaginare a priori.

Affrontiamo ora un esempio che ci aiuti a provare queste affermazioni. Lo spunto ci viene suggerito dallo stesso Stephen Hawkings: la legge della gravitazione di Newton. Ci daremo come scopo quello di descrivere dapprima il moto della Terra attorno al Sole e poi aggiungeremo anche la Luna. Il modello completo della gravitazione di Newton poggia su alcune leggi di valore propedeutico ed enuncia quindi le relazioni quantitative. Ma andiamo per ordine; la prima legge dice che un corpo rimane nel suo stato di riposo o di moto uniforme su una linea retta fino a quando non interviene una forza esterna che modifica la situazione. Osservando il moto dei corpi celesti ci si accorge che essi si spostano su traiettorie ellittiche: ci deve quindi essere una forza esterna che modifica lo stato delle cose. La seconda legge dice che la quantità di moto di un oggetto (massa moltiplicata per la velocità) cambia in maniera proporzionale alla forza applicata e nella stressa direzione in cui viene applicata la forza stessa. La terza legge dice che ogni azione produce una reazione opposta e contraria. E infine, la legge di gravitazione universale ingloba in essa tutte le precedenti e ne da' anche una quantificazione:

$$F = \frac{Gm_1\, m_2}{r^2}$$

Essa dice che quello che devia i corpi dal loro stato di riposo o di moto rettilineo é la forza gravitazionale che scaturisce dal fatto che i corpi possiedono una massa. Questa forza agisce lungo la retta che unisce i corpi e fa cambiare continuamente la quantità di moto. Inoltre, la forza che agisce su un corpo dovuta alla presenza dell'altro é identica a quella che agisce sul secondo dovuta alla presenza del primo. Questa forza complessiva é proporzionale al prodotto delle due masse ed inversamente proporzionale al quadrato della distanza che le separa. Le forze gravitazionali si possono sommare per ottenere una forza complessiva e quindi questo modello é per sua natura estensibile ad un numero di corpi qualsiasi. Cominceremo con il vedere come questo modello si applica al caso più semplice di due corpi: il Sole e la Terra. Nell'implementare il nostro modello dobbiamo tenere conto di alcune questioni: il moto é tridimensionale, bisogna immaginarsi una maniera algoritmica di usare il modello, dobbiamo scegliere il livello di accuratezza dei valori scelti come valori iniziali e dobbiamo trovare un buon valore per il passo temporale al quale intendiamo effettuare i calcoli. Il fatto che il moto avvenga in tre dimensioni non ci impressiona troppo poiché sappiamo che le forze possono essere sommate per ottenere una risultante e quindi possiamo scomporre la nostra forza gravitazionale lungo i tre assi cartesiani principali (x, y e z) e comunque la forza totale risultante sarà corretta. Se pensiamo a cosa che sia necessario fare per effettuare i nostri calcoli in maniera corretta ci accorgiamo che é necessario partire dal determinare la velocità alla quale si muovono i corpi per poi risalire alle posizioni. Così la sequenza di calcoli é la seguente:

$$v = v_0 + F\Delta\, t$$
$$p = p_0 + v\Delta\, t$$

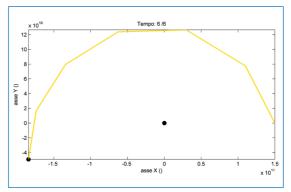



Fig. 1a: Delta t = 30 giorni, 6 iterazioni (180 giorni complessivi).

Se procediamo iterativamente in questo modo possiamo avanzare nel tempo fino a raggiungere un appropriato orizzonte temporale che descriva il fenomeno appropriatamente. In questa maniera possiamo prevedere posizione e velocità dei corpi in funzione del tempo. Questo procedimento é lineare e presuppone che la traiettoria seguita dai corpi nel tempo delta t sia rettilinea; cosa che é evidentemente non corretta poiché sappiamo invece che i corpi celesti che interagiscono gravitazionalmente tra loro percorrono traiettorie curvilinee. Se pensiamo di rimpicciolire *"delta t"* possiamo minimizzare a piacimento l'errore che commettiamo ma comunque commetteremo sempre un certo errore; il problema sta nel renderlo trascurabile. Se proviamo ad iterare per un numero di giorni pari a 180 con un passo *"delta t"* di un mese (30 giorni) per iterazione e quindi con un *"delta t"* pari a 10 giorni, raggiungiamo i risultati delle Fig. 1a e 1b.
Vediamo ad occhio nudo che essi sono qualitativamente differenti e se guardiamo con maggiore at-

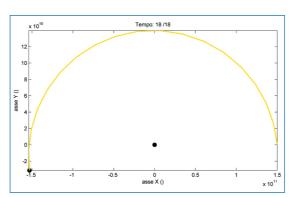



Fig. 1b: Delta t = 10 giorni, 18 iterazioni (180 giorni complessivi).

Modello

Nell'implementare il nostro modello dobbiamo tenere conto di alcune questioni: il moto é tridimensionale, bisogna immaginarsi una maniera algoritmica di usare il modello, dobbiamo scegliere il livello di accuratezza dei valori scelti come valori iniziali e dobbiamo trovare un buon valore per il passo temporale al quale intendiamo effettuare i calcoli.

MATLAB

[1] Data AspectRatio

La proprietà *DataAspectRatio* regola l'enfatizzazione da dare ai rispettivi assi cartesiani.

[2] View

La funzione *view* imposta un angolo sotto il quale si decide di vedere la scena.

[3] Drawnow

Drawnow impone di eseguire un render della figura grafica, altrimenti MATLAB visualizza i grafici soltanto al termine dell'elaborazione.

tenzione scopriamo che anche numericamente vi sono differenze sostanziali. Per esempio la coordinata *y* massima nell'uno é pari a poco meno di $1.3\ 10^{11}$ mentre nel secondo é pari a quasi $1.4\ 10^{11}$. Matematicamente parlando stiamo trattando un problema del valore iniziale. Qui risolviamo le equazioni del moto integrandole numericamente piuttosto che andare alla ricerca di una soluzione analitica. Il metodo descritto prende il nome di *"Metodo di Eulero"*. Un grande numero di problemi fisici possono essere risolti usando metodi basati su questa tecnica. Di conseguenza, un grande ammontare di sforzi sono stati profusi dalla comunità dei matematici per sviluppare versioni veloci ed accurate di queste tecniche. Fino ad ora ci siamo concentrati sul modello matematico ma ad un certo punto dobbiamo però arrivare a chiederci di quanta accuratezza abbiamo bisogno. Questo dipende da un certo numero di fattori: cosa vogliamo descrivere, la disponibilità di dati e misure di accuratezza sufficiente e la disponibilità di sistemi di calcolo (hardware e software) adatti a sopportare il peso computazionale. Qui il nostro obbiettivo é quello di comprendere quale sia l'interazione tra due corpi celesti e sperimentare alcune situazioni notevoli che conosciamo e che useremo come casi di test qualitativo. Questo ci servirà per fare evolvere il nostro modello fino a trattare fenomeni molto complessi. I dati reperibili sui corpi del sistema solare sono di qualità sufficiente e sono da lungo tempo al vaglio degli astronomi che ne controllano l'accuratezza con continuità. Per quanto riguarda i sistemi di calcolo abbiamo oggi a disposizione software di calcolo come MATLAB che ci consentono quello che ci era precluso solo pochi decenni or sono. La prima simulazione che tenteremo di eseguire é relativa al moto terrestre attorno al Sole. Per farlo costruiamo il nostro solito script (*simulplanet.m*) che contiene la definizione delle variabili caratteristiche del problema e la chiamata al programma che effettua la simulazione vera e propria. Concentriamoci ora su quello che é il vero e proprio motore di calcolo della nostra simulazione (*planet.m*). La prima operazione che facciamo é quella di inizializzare la visualizzazione grafica e di predisporre l'ambiente nel quale mostreremo il risultato dei calcoli.

```
function planet(massa_tgt,x_tgt,y_tgt,z_tgt,
                                    vx_tgt,vy_tgt,vz_tgt,...
        massa_int,x_int,y_int,z_int,vx_int,vy_int,vz_int,...
                                        dT_iter,iteraz, dT);
% PLANET Simulazione dell'interazione tra pianeti

% Inizializzazione del campo grafico di disegno
f=figure;
set(f,'Renderer','Zbuffer',...
    'Color',[.6 .6 .6]);
[1] axh=axes('Box','on',...
    'DrawMode','Fast',...
    'DataAspectRatio',[1, 1, 1],...
    'Position',[0.1 0.1 0.8 0.8],...
    'Color',[0 0 0],...
    'XColor',[1 1 1],...
    'YColor',[1 1 1],...
    'ZColor',[1 1 1]);
[2]
view([-45 45]);
axis equal
xlabel('asse X ()');
ylabel('asse Y ()');
zlabel('asse Z ()');
```

Definiamo quindi la *costante di gravitazione universale (G)* secondo il sistema internazionale *MKS* (metri, chilogrammi, secondi). Adottiamo subito una terminologia che ci sarà più chiara quando estenderemo la trattazione alle galassie ma che si presta bene anche in questo caso. Chiamiamo il corpo celeste che ha maggiore massa e che risiede inizialmente più vicino al centro di rotazione del sistema (centro di massa), *"bersaglio"*, mentre il secondo lo chiameremo *"intruso"*. Considereremo l'*intruso* come il corpo che con il suo campo gravitazionale va a perturbare la quiete o il moto uniforme del *bersaglio*.

Nel nostro codice posizioniamo i due punti che rappresentano *bersaglio* e *intruso* nelle rispettive posizioni.

```
% Inizializzazione variabili
G = 6.67259e-11; % N * m^2 / kg^2

hold on

% Centri del TARGET e dell'INTRUDER
bersaglio = plot3(x_tgt,y_tgt,z_tgt,'Marker','.','
                            MarkerSize',25,'Color',[1 1 1]);
intruso = plot3(x_int,y_int,z_int,'Marker','.','
                            MarkerSize',25,'Color',[1 1 1]);
[3]
drawnow
axis equal;
```

Ecco ora che viene la parte più algoritmica del programma. Notiamo subito che abbiamo due cicli *"for"* che ci consentono di minimizzare la rappresentazione grafica che é sempre dispendiosa come tempo e risorse di memoria. Le due variabili che comandano i cicli sono, dal più interno, *"dT_iter"* e *"iteraz"*.

La variabile *"dT_iter"* fa eseguire il calcolo delle velocità e delle posizioni un certo numero di volte e solo allora consente di eseguire uno step di visualizzazione.

La variabile *"iteraz"* fa ripetere questo processo *n* volte. Avendo stabilito un *"delta t"* appropriato (vedi *simulplanet.m*) possiamo calcolare l'orizzonte temporale della simulazione come prodotto tra *delta t*, *dT_iteraz* e *iteraz*.

```
[4]
for j=1:iteraz
   for k=1:dT_iter
      % Calcolo della velocita' del centro del pianeta
                                                  TARGET
      opt1 = x_tgt - x_int;
      opt2 = y_tgt - y_int;
      opt3 = z_tgt - z_int;
      opt4 = (opt1^2 + opt2^2 + opt3^2 + 1e9)^1.5;
      vx_tgt = vx_tgt - G*massa_int*opt1/opt4*dT;
      vy_tgt = vy_tgt - G*massa_int*opt2/opt4*dT;
      vz_tgt = vz_tgt - G*massa_int*opt3/opt4*dT;
      % Calcolo della velocita' del centro del pianeta
                                                INTRUDER
      vx_int = vx_int + G*massa_tgt*opt1/opt4*dT;
      vy_int = vy_int + G*massa_tgt*opt2/opt4*dT;
      vz_int = vz_int + G*massa_tgt*opt3/opt4*dT;
      % Aggiornamento della posizione del centro
                                     del pianeta TARGET
      x_tgt = x_tgt+vx_tgt*dT;
      y_tgt = y_tgt+vy_tgt*dT;
      z_tgt = z_tgt+vz_tgt*dT;
      % Aggiornamento della posizione del centro
                                   del pianeta INTRUDER
      x_int = x_int+vx_int*dT;
      y_int = y_int+vy_int*dT;
      z_int = z_int+vz_int*dT;
      % Disegno della traiettoria dei centri dei pianeti
      line([x_tgt-vx_tgt*dT x_tgt],[y_tgt-vy_tgt*dT
           y_tgt],[z_tgt-vz_tgt*dT z_tgt],'Color',[1 1 0],
                                         'linewidth',2);
      line([x_int-vx_int*dT x_int],[y_int-vy_int*dT
            y_int],[z_int-vz_int*dT z_int],'Color',[1 1 0],
                                         'linewidth',2);
   end
   % Varizione della posizione dei centri dei pianeti
   set(bersaglio,'XData',x_tgt,'YData',y_tgt,'ZData',z_tgt);
   set(intruso,'XData',x_int,'YData',y_int,'ZData',z_int);
   % Calcolo del tempo trascorso
   tempo=(j-1)*dT_iter+k;
[5]
   % Aggiornamento delle visualizzazione
   title(['Tempo: ',num2str(tempo),'
                          /',num2str(iteraz*dT_iter)]);
   drawnow
   axis equal;
end
```

Le espressioni matematiche qui utilizzate sono leggermente differenti dalla forma canonica che abbiamo visto sopra. Questo avviene perché siamo in uno spazio tridimensionale e quindi dobbiamo calcolare la componente della forza generata da ogni singolo corpo per mezzo del suo potenziale gravitazionale Kepleriano (quindi dipendente soltanto dalla sua massa) lungo ogni asse cartesiano (x, y e z). Dato un potenziale Kepleriano definito come $P(r) = -GM/r$ ("M" e' la massa del corpo ed "r" è la distanza a cui si considera il potenziale), la forza che agisce ad una distanza "r" é $F(r) = -GM/r^2$. In coordinate cartesiane tridimensionali la forza ha componenti Fx, Fy, Fz dove $Fx = -2GMx/r^3$ con espressioni simili nelle altre due componenti. Notiamo ancora che il termine in r^3 non é scritto esattamente così ma compare un termine aggiuntivo. Se proviamo a tracciare la curva delle velocità di un oggetto che percorre un'orbita attorno ad un corpo dotato di sufficiente massa in funzione della distanza dal corpo, in accordo con la forma canonica di Newton notiamo che la velocità cresce indefinitamente mano a mano che ci avviciniamo. Questo termine rende invece la simulazione più realistica "bloccando" la crescita indefinita della velocità all'avvicinarsi dei corpi. Ovviamente si assume che le masse siano puntiformi, vale a dire concentrate nel centro dei corpi celesti.

Vale qui la pena di notare due porzioni del codice che sono interessanti. Le linee di codice che contengono le funzioni *"line"* creano due linee, l'una che traccia il movimento del bersaglio, l'altra quello dell'intruso. Appena si esce dal ciclo più interno si incontrano due funzioni *"set"*; abbiamo visto pochi momenti fa che le variabili *"bersaglio"* e *"intruso"* contengono gli handle grafici dei punti che rappresentano il Sole e la Terra. Queste vengono usate per cambiare le loro proprietà di posizione; trattandosi di punti, le proprietà *"Xdata"*, *"Ydata"* e *"Zdata"* contengono uno scalare. Variando questo scalare con le variabili appena calcolate forziamo MATLAB a rappresentare i punti nelle loro nuove posizioni. Si tratta di un metodo particolarmente veloce poiché non implica la creazione ex-novo del grafico ma soltanto una sua modificazione. Le rimanenti istruzioni producono ancora qualche marginale operazione di maquillage del grafico e quando anche il ciclo più esterno si completa termina anche la simulazione. A questo punto viene lasciato visibile il grafico risultante di Fig. 2.

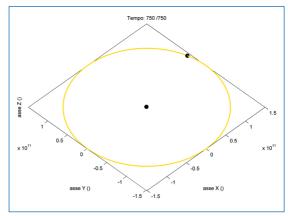



Fig. 2: **Orbita terrestre.**

Possiamo ora provare ad esplorare i risultati della simulazione rispetto a tre fattori che sono di qualche interesse. Se facciamo uno zoom sul Sole (Fig. 3) vediamo che la sua traiettoria ha assunto la forma di un tre ed esso si trova ora ad una distanza pari a cir-

[4] Opt

Le variabili *opt...* vengono usate per minimizzare i tempi di calcolo e per comodità di scrittura del codice.

[5]

La stringa usata come titolo del grafico viene aggiornata ad ogni step di visualizzazione per dare informazioni sull'avanzamento dell'elaborazione.

MATLAB

Sul Web

Getting Started with MATLAB
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

Using MATLAB
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/using_ml.pdf

Using MATLAB Graphics
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/graphg.pdf

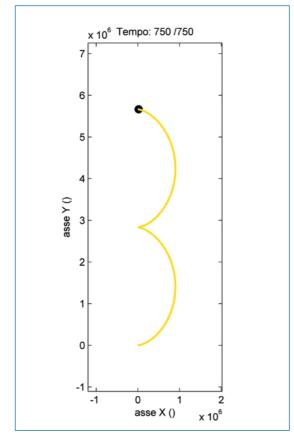



Fig. 3: Spostamento del Sole dovuto a due orbite terrestri.

ca 6000 km dal punto di partenza. Per comprendere bene di cosa si tratti proviamo a considerare il fatto che il diametro del Sole é pari a *1390000 km*; é come dire che se il Sole fosse una pallina da tennis questa si sarebbe spostata di circa 0.3 millimetri. Proviamo ora a zoomare sulla Terra e vediamo che essa non percorre sempre la stessa traiettoria. Questo é principalmente dovuto a due fattori: il Sole si muove anche se di poco e la velocità iniziale é stata calcolata in un modo estremamente semplicistico. In pratica la velocità é stata calcolata come lo spazio ottenuto dalla circonferenza di raggio pari al raggio medio dell'orbita terrestre diviso il tempo di un'orbita (365.26 giorni). In effetti questa é un'assunzione poco precisa poiché avremmo dovuto assumere un'orbita ellittica. In realtà il nostro scopo era quello di testare il modello per vedere se era in grado di replicare il fenomeno naturale. Se avessimo il dubbio che esso possa simulare traiettorie ellittiche basta provare ad aumentare la velocità iniziale del 10% ed otterremmo subito un'orbita decisamente ellittica. Il nostro modello descrive bene il moto di due corpi: tentiamo ora di introdurne un terzo che, nel nostro caso, é facile individuare nella Luna. Se diamo un'occhiata ai file *"simulsatel.m"* e *"satel.m"* notiamo poche modifiche sostanziali.
A parte il codice necessario per trattare anche i dati della Luna. Notiamo una sola importante modifica: il calcolo delle forze (e relative velocità) vede ora comparire un ulteriore termine che ci consente di calcolare le interazioni applicate al terzo corpo e le forze da questo generate sugli altri due. Siamo ora pronti a lanciare un'ulteriore simulazione. Il risultato é particolarmente buono rispetto ad un nuovo fenomeno che vediamo verificarsi sotto i nostri occhi. La traiettoria della Terra é perturbata dalla Luna; infatti, se osserviamo attentamente vediamo che l'arco descritto dalla Terra non é più così regolare ma presenta degli ondeggiamenti che sono caratteristici di questo tipo di interazioni gravitazionali. Questo effetto é apprezzabile soprattutto guardando l'orbita di taglio; è ora facile vedere che essa non giace più su di un piano ma viene deformata dall'azione della Luna che ha un'orbita inclinata di circa 5 gradi rispetto al piano dell'eclittica (vedi Fig. 4).

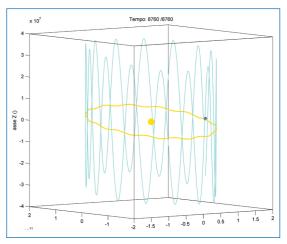



Fig. 4: Deformazione dell'orbita terrestre dovuta alla presenza della Luna (l'asse z e' stato enfatizzato per amplificare l'effetto e renderlo maggiormente visibile).

CONCLUSIONI

La trattazione di modelli matematici é un'attività' di cui é imbevuta tutta la tecnologia moderna. Essa ha un'importanza cruciale per individuare le caratteristiche dei fenomeni e per essere in grado di controllarli adeguatamente. Nei prossimi numeri continueremo il nostro cammino e utilizzeremo i concetti qui esposti per estenderli a situazioni più complesse e rilevanti. Inoltre, invito tutti coloro i quali vogliano contribuire con suggerimenti e idee a contattarmi direttamente per mezzo dell'indirizzo di posta elettronica citato sotto.
Per maggiori informazioni sui prodotti della famiglia MATLAB potete consultare il sito di The MathWorks (*www. mathworks.it*). Suggerisco a tutti di dare un'occhiata ad una porzione del sito web chiamato *"MATLAB Central"* (*www.mathworks.com/matlabcentral/*). Esso riporta innumerevoli esempi di uso di MATLAB in una varietà molto vasta di discipline tecnico scientifiche.

Fabrizio Sara
(fabrizio.sara@mathworks.it)

Visual Basic

Applicazioni con interfacce avanzate.

Effetti speciali sui Form

In questo appuntamento utilizzeremo le API di Windows per aggiungere nuove features ai Form, in particolar modo vedremo come realizzare delle form trasparenti e di dimensioni e forme non necessariamente standard.

Le API *(Application Programming Interface)* di Windows sono i "blocchi" con i quali si costruiscono le applicazioni Windows-based. Le funzioni dell'API sono supportate sia dai sistemi operativi a 32-bit che dai sistemi operativi a 64-bit, in parole povere sono utilizzabili sia da Windows 98 che da Windows Server 2003. Le API sono tante e possono essere usate per svariati scopi. Esse sono organizzate in categorie, per esempio la categoria che a noi principalmente interessa in questo appuntamento è la categoria *regione (region)* che contiene le funzioni per creare e gestire delle regioni (rettangoli, poligoni ecc.) sullo schermo.
Le funzioni della categoria regione, per esempio, possono essere usate per rendere i form trasparenti, per dare alle finestre una forma particolare (poligonale, ovale, con buchi ecc.), mentre le funzioni di altre categorie, come *Windows Function* e *Brush Functions*, possono essere usate per dotare la finestra di effetti speciali (compressione, espansione, comparsa della finestra dall'alto ecc.) , per capirci un po' come siamo abituati con ambienti quali Flash, Power Point ecc. Noi, naturalmente, descriveremo solo alcune funzioni dell'API e come applicazione realizzeremo un visualizzatore di immagini simile a quello presente in Windows XP. In particolare in questo appuntamento analizzeremo i seguenti argomenti:

1. la sintassi di alcune funzioni dell'API;
2. descrizione sommaria degli oggetti della libreria *MsForms 2.0*;
3. realizzazione di una finestra di forma rotonda;
4. introduzione all'applicazione *"Visualizzatore Immagini"*.

LE FUNZIONI DEFINITE NELLA CATEGORIA REGION

Come già in precedenza accennato, la categoria *Region* include le funzioni dell'API che permettono di manipolare i Form. Le funzioni principali della categoria *Re-*

Funzione	Descrizione sommaria
CombineRgn	Combina due regioni e memorizza il risultato in una terza regione
CreateEllipticRgn	Crea una regione ellittica
CreateEllipticRgnIndirect	Crea una regione ellittica definita attraverso una struttura Rect (rettangolo)
CreatePolygonRgn	Crea una regione poligonale
CreatePolyPolygonRgn	Crea una regione fatta di una serie di poligoni
CreateRectRgn	Crea una regione rettangolare
CreateRectRgnIndirect	Crea una regione rettangolare definita attraverso una struttura Rect
CreateRoundRectRgn	Crea una regione rettangolare con spigoli rotondi
EqualRgn	Controlla se due regioni sono uguali
ExtCreateRegion	Crea una regione in base ad un altra regione e a dei criteri di conversione
FillRgn	Riempie una regione usando uno specifico pennello
FrameRgn	Disegna il bordo di una regione usando uno specifico pennello
InvertRgn	Inverte il colore di una regione
OffsetRgn	Muove una regione di uno specifico Offset
PaintRgn	Colora una regione usando il pennello selezionato
PtInRegion	Determina se un punto è in una regione
RectInRegion	Determina se la parte di un rettangolo è dentro il bordo di una regione
SetPolyFillMode	Specifica il tipo di riempimento per un poligono
SetRectRgn	Converte una regione in una regione rettangolare

Tab. 1: **Funzioni definite all'interno della categoria Region**

API

Le caratteristiche dell'API sono racchiuse in delle DLL (*dynamic-link library*) ovvero file che contengono procedure, funzioni ecc. Esse sono caricati e collegati alle applicazioni durante l'esecuzione. Il modo migliore per includere nei progetti Visual Basic le caratteristiche dell'API è utilizzare l'applicazione *Apiload.exe* (Visualizzatore API).

Visual Basic

gion sono brevemente descritti in Tab1. Una regione può essere un rettangolo, un poligono o un'ellisse (o una combinazione di due o più di queste forme). Le regioni possono essere riempite, invertite, colorate, incorniciate e usate per individuare la posizione del cursore. Le funzioni che tra poco descriveremo - e la maggior parte di quelle che permettono di gestire le caratteristiche grafiche del sistema - appartengono alla libreria *GDI32.dll (Graphical Device Interface)*.
La *GDI32.dll* comunica con le applicazioni *Windows-Based* attraverso la *GDI (Windows Graphics Device Interfacce)* e con i driver della stampante attraverso la *DDI (Device Driver Interfacce)*.
Le funzioni che utilizzeremo nei nostri esempi sono le seguenti:

```
Public Declare Function CreateRectRgn Lib "gdi32"
        (ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As
                    Long, ByVal Y2 As Long) As Long
```

Come specificato evidenziato in Tab.1 questa funzione permette di creare una regione rettangolare in base ai valori dei seguenti parametri: *X1, Y1, X2, Y2*. Essi rappresentano rispettivamente le coordinata *x* e *y* del punto superiore sinistro e del punto inferiore destro del rettangolo che verrà disegnato.

```
Public Declare Function CreateEllipticRgn Lib "gdi32"
        (ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As
                    Long, ByVal Y2 As Long) As Long
```

Anche per questa funzione le coordinate rappresentano un rettangolo, ovvero quello che contiene l'ellisse che sarà disegnata, quindi il significato dei parametri è analogo a quello della funzione analizzata in precedenza:

```
Public Declare Function CombineRgn Lib "gdi32" (ByVal
        hDestRgn As Long, ByVal hSrcRgn1 As Long, ByVal
                        hSrcRgn2 As Long, ByVal
nCombineMode As Long) As Long
```

Questa funzione crea una regione combinandone altre due, *hSrcRgn1* è l'handle della prima regione, *hSrcRgn2* è l'handle della seconda regione, *hDestRgn* rappresenta l'handle della regione risultato. Il parametro *nCombineMode* stabilisce come saranno combinate le due regioni, a tal proposito di utile supporto può essere la Tab. 2.

Costante	Valore	Effetto sulla terza regione
RGN_AND	1	La terza regione è l'intersezione delle altre due
RGN_COPY	2	Crea una copia della regione identificata da *hrgnSrc1*
RGN_DIFF	3	La terza regione è formata dalle parti di *hrgnSrc1* che non sono parti di *hrgnSrc2*
RGN_OR	4	La terza regione è l'unione delle due regioni
RGN_XOR	5	La terza regione è l'unione delle due regioni eccetto le parti sovrapposte

Tab. 2: **Costante che specifica la modalità di visualizzazione della terza regione.**

Nei paragrafi che seguiranno descriveremo alcune funzioni che non appartengono alla categoria *Region* ma che sono di fondamentale importanza per la gestione delle finestre che implementeremo.

LA FUNZIONE CHE DISEGNA LA FINESTRA

```
Public Declare Function SetWindowRgn Lib "user32"
        (ByVal hWnd As Long, ByVal hRgn As Long, ByVal
                    bRedraw As Boolean) As Long
```

Questa funzione, a differenza delle altre, è classificata nella categoria "tracciare e dipingere" (*Painting* e *Drawing*). Essa imposta la *Window Region* (regione di una finestra) cioè l'area dentro la finestra in cui il sistema permette il *drawing*. Naturalmente il sistema non mostra le parti di finestra fuori della *Window Region*.
Il significato dei parametri è il seguente: *hWnd* identifica la finestra che è stata impostata; *hRgn* identifica la regione interna alla finestra che si sta definendo (se *hRgn* è *null* anche la *Window Region* assumerà valore *null*); *bRedraw* è un valore di tipo Boolean che specifica se il sistema ridisegnerà la finestra dopo averla impostata, solitamente, se la finestra è visibile, questo valore è impostato su *True*.

LE FUNZIONI PER CONTROLLARE IL MOUSE

Allorché si disegnano finestre di forma diversa da quella nativa, per controllare i messaggi di input provenienti dal mouse, conviene usare delle funzioni dell'API. Questo perché in generale, nelle finestre di forma non canonica, la barra del titolo (la porzione d'interfaccia in cui sono presenti i controlli che permettono lo spostamento e il ridimensionamento del form) viene nascosta. Le funzioni che possono essere utilizzate a tal fine sono le seguenti:

```
Private Declare Function SendMessage Lib "user32" Alias
        "SendMessageA" (ByVal hWnd As Long, ByVal wMsg
                    As Long, ByVal wParam As Long,
lParam As Any) As Long

Private Declare Function ReleaseCapture Lib "user32" ()
                                            As Long
```

La *ReleaseCapture* appartiene alla categoria *"Mouse Input"*. Questa permette di catturare gli input provenienti dal mouse. La *ReleaseCapture* è senza parametri.
La *SendMessage*, invece, appartiene alla categoria *Messaggi (Message and Message Queue)*, essa invia un messaggio ad una finestra o ad un insieme di finestre ed aspetta che questo venga elaborato. I parametri della

Metodo ScaleX e ScaleY

***ScaleX* e *ScaleY* servono per convertire il valore dell'ampiezza e dell'altezza di un oggetto in base alle unità di misura specificate attraverso la proprietà *ScaleMode*. La sintassi di ScaleX è la seguente: *oggetto.ScaleX* (larghezza, da-scala, a-scala), il secondo ed il terzo parametro devono essere i valori ammessi per la proprietà ScaleMode.**

funzione sono i seguenti:

- **hWnd:** l'identificatore della finestra di destinazione;
- **wMsg:** il messaggio da spedire;
- **Wparam e Lparam:** rispettivamente il primo e il secondo parametro del messaggio.

Infine per eliminare gli oggetti creati useremo la seguente funzione.

```
Public Declare Function DeleteObject Lib "gdi32"
                    (ByVal hObject As Long) As Long
```

Questa funzione elimina l'oggetto associato ad un elemento grafico (*pen*, *brush*, *font*, *bitmap*, *region*, ecc.).

LA LIBRERIA MSFORM2

La *MsForm2 library (FM20.dll)* contiene i controlli che ampliano le caratteristiche di alcuni elementi nativi di Visual Basic come il *Frame*, il *CommandButton*, l'image ecc. Per esempio, sia il *Frame* che il *CommandButton* di *MsForm2* possono essere riempiti con un'immagine o con un colore. I controlli di *MsForm2*, rispetto ai controlli nativi, non hanno una finestra propria (sono *WindowLess*) per questo non sono adatti per alcune tecniche di programmazione (come il *subclassing*).

LA PRIMA FINESTRA DI FORMA ROTONDA

Per applicare i concetti esposti implementeremo un semplice esempio; in particolare costruiamo il form rotondo mostrato in Fig.1.
Per far ciò dobbiamo creare un nuovo progetto con un form e su di esso dobbiamo inserire una label (*"Entra nel mio negozio"*), un oggetto *Shape* (di tipo *circle*) e un

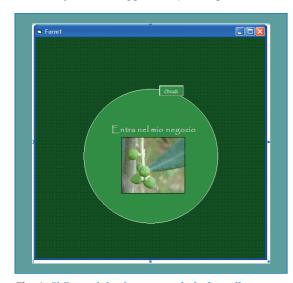



Fig. 1: **Il Form del primo esempio in fase di progettazione.**

Fig. 2: **Il Form durante l'esecuzione, sullo sfondo il desktop di Windows XP.**

CommandButton della libreria *MsForm2*. I colori e le immagini del form e dei suoi elementi li impostiamo attraverso le finestre delle proprietà. Nella parte dichiarativa del form dobbiamo inserire, tra l'altro, la definizione delle funzioni descritte in precedenza, cioè di *CreateEllipticRgn*, *CreateRectRgn*, *CombineRgn*, *SetWindowRgn*, *SendMessage*, *ReleaseCapture*, *DeleteObject*. Successivamente alla definizione delle funzioni dobbiamo inserire il seguente codice:

```
Dim iden As Long
Private Const RGN_DIFF = 3
Private Const RGN_AND = 1

Private Function Crearegione() As Long
 Dim RegCombinate As Long
 Dim NuovaRegione As Long
 Dim regForm As Long
 regForm = CreateRectRgn(0, 0, Width, Height)
 RegCombinate = CreateRectRgn(0, 0, 0, 0)
 iden = CombineRgn(RegCombinate, RegCombinate,
                                   regForm, RGN_DIFF)
 NuovaRegione = CreateEllipticRgn(74, 108, 464, 498)
 iden = CombineRgn(RegCombinate, RegCombinate,
                            NuovaRegione, RGN_AND)
 Crearegione = RegCombinate
End Function
```

La funzione *Crearegione* serve per dare alla finestra la forma circolare. La *Crearegione* per realizzare ciò prima combina (con *nCombineMode =RGN_DIFF*) una regione che copre completamente il form con una regione vuota e poi interseca (con *nCombineMode =RGN_AND*) il risultato con una regione circolare. Facciamo notare che quando si imposta la differenza (*RGN_DIFF*) tra una regione vuota e una piena resta solo la piena. Le coordinate dei due punti, necessari per creare un rettangolo che copra tutto il form, sono stati posti sullo spigolo superiore (*0,0*) e sullo spigolo inferiore destro del form (*Width*, *Height*).
Attenzione, però, a non modificare i valori di default delle proprietà *Scale* del Form. I valori dei parametri di

Visual Basic

Varie proprietà

Le proprietà *ScaleMode, ScaleHeight, ScaleWidth, ScaleLeft* e *ScaleTop* servono per specificare il tipo e la dimensione, in termine di unità di misura, del sistema di coordinate di un oggetto. *ScaleHeight* imposta, o fornisce, il numero di unità per l'asse verticale (*ScaleWidth* per quello orizzontale), per esempio *ScaleHeight= 50* imposta a 50 le unità massime per l'asse verticale (invece di n Twips impostati di default). Scale Mode, invece, imposta o restituisce il tipo di unità di misura utilizzata per un oggetto (0=definita dall'utente, 1=Twip..., 7= centimetri ecc.).

Visual Basic

CreateEllipticRgn, cioè i punti (74, 108) e (464, 498), sono stati valutati in modo che l'ellisse tracciata sia una circonferenza che circondi il controllo *Shape* presente sul form. La regione della finestra viene disegnata, mediante la funzione *SetWindowRgn*, nell'evento *Form_Load*:

```
Private Sub Form_Load()
   iden = SetWindowRgn(hWnd, Crearegione, True)
End Sub
```

Notate che la *SetWindowRgn* richiama la *CreaRegione* che restituisce l'identificatore della regione da disegnare.

CATTURIAMO I MOVIMENTI DEL MOUSE

Se ricordate bene, in precedenza abbiamo nascosto la barra del titolo, ragion per cui, ora, per spostare il form sullo schermo utilizziamo la *ReleaseCapture* e la *SendMessage* (con degli opportuni parametri) che inseriamo nella *Form_MouseDown*. I parametri della *SendMessage* saranno spiegati nel successivo appuntamento.

```
Private Sub Form_MouseDown(Button As Integer,
                     Shift As Integer, X As Single, Y As Single)
 ReleaseCapture
 SendMessage Me.hWnd, &HA1, 2, 0&
End Sub
```

Il codice relativo al bottone *Chiudi* è il seguente:

```
Private Sub Chiudi_Click()
 Unload Me
End Sub
```

IL VISUALIZZATORE IMMAGINI

In questo paragrafo accenniamo all'applicazione *"Visualizzatore Immagini"* che completeremo nel successivo appuntamento; si tratta di un visualizzatore di immagini simile a quello incluso nei sistemi operativi Microsoft (per esempio Windows XP). L'interfaccia della versione base dell'applicazione è mostrata in Fig. 3, essa presenta una barra di comando costituita da una serie di pulsanti che permettono di sfogliare le immagini, di proiettarle, di ricercarle sul computer o su internet, d'ingrandirle ecc. Inoltre c'è d'aggiungere che le varie parti dell'applicazione compaiono e scompaiono con l'effetto espansione/compressione e che … analizzeremo in dettaglio nel prossimo appuntamento!
In buona sostanza abbiamo implementato un'applicazione con delle caratteristiche "che suscitano invidia" al visualizzatore di Windows XP.

Fig. 3: **Il Form del visualizzatore di immagini in fase di progettazione.**

IL VISUALIZZATORE DI WINDOWS XP

Il nome completo del visualizzatore di Windows XP è "Visualizzatore Immagini e fax Windows" visto che con esso, oltre che poter gestire immagini dai formati più disparati, è possibile gestire i documenti di Microsoft Console Servizio Fax e le immagini prodotte con periferiche esterne come fotocamere digitali, scanner ecc. Il visualizzatore di Immagini e Fax è facilmente importabile su un form Visual Basic (però non è altrettanto facile controllarlo).

Fig. 4: **Il visualizzatore immagini di Windows XP.**

La libreria da importare nei progetti è "Preview 1.0 Type Library". Microsoft Console Servizio Fax è un tool che potete trovare nel CD di Windows XP Professional.

Librerie e tool

La libreria da importare nei progetti è *"Preview 1.0 Type Library"*. *Microsoft Console Servizio Fax* è un tool che potete trovare nel CD di Windows XP Professional.

CONCLUSIONI

La finestra di forma rotonda che abbiamo presentato può essere la maschera principale di una vostra applicazione, naturalmente essa va corredata con i pulsanti che permettono di avviare le altre parti dell'applicazione.
Il progetto che presenteremo nel successivo appuntamento è un Visualizzatore di Immagini con molte sorprese. In parallelo illustreremo come attivare, da un progetto Visual Basic, il Visualizzatore di Immagini di Windows XP.

Massimo Autiero

Ambienti renderizzati parte quarta

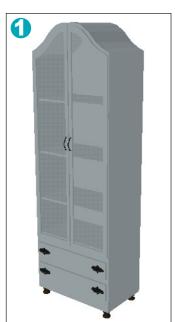
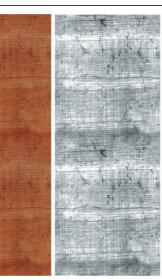
▶ Lightwave

Roberto Lombardo

Ultimiamo il nostro modello con il texturing del mobilio e settando i parametri per il rendering finale.

Con questo quarto e ultimo tutorial, concluderemo il nostro modello ultimando la texturizzazione del mobilio e la definizione della scena finale per il rendering. Questa serie di tutorial mira a fornire una conoscenza di base sul procedimento costruttivo di un modello/scena di questo genere, realizzando il prodotto finale tramite una divisione del lavoro in passaggi intermedi. Parliamo quindi della definizione/modellazione della stanza base "grezza", dei suoi particolari, del mobilio e oggetti di contorno. Stesso procedimento per la fase di texturing. Nel nostro caso la stanza è molto semplice, nei progetti più complessi questa metodologia di lavoro ci verrà in aiuto per un'ottima organizzazione del processo produttivo, specie se bisogna realizzare molti ambienti. In questo ultimo tutorial vedremo nel dettaglio la texturizzazione della credenza, della poltroncina, la mensola, i quadri e le fonti luminose (lampadario e appliques). Completato questo passaggio imposteremo le luci in scena ed eseguiremo il rendering finale in 2 diverse riprese di camera. Non ci resta che iniziare con…

▼1-2-3 La credenza

In figura 1 possiamo vedere la credenza creata nel numero precedente, un oggetto molto semplice che contribuirà a rendere più realistica la stanza che stiamo realizzando. A quest'oggetto applicheremo una texture di tipo legno che verrà "mappata" in modo planare sui vari assi del nostro oggetto: **x,y** e **z**.. Non occorre, in questo caso, rendere la texture tailizzabile, possiamo realizzarla direttamente proporzionale alle dimensioni della credenza, in modo da poter aggiungere dettagli e "sporco" in alcuni punti precisi del modello. In figura 2 possiamo vedere la texture come viene applicata sul modello.
In figura 3 abbiamo le texture impiegate in questo modello e un render di prova del solo oggetto, possiamo notare 2 versioni della stessa texture, una a colori e l'altra in toni di grigio; quest'ultima versione, servirà per i canali di specularità, diffusione e bump della superficie, al fine di rendere più realistico il materiale da noi creato, in questo caso il legno. Ogni texture che impiegheremo in questa scena sarà quindi in duplice versione, sia a colori che in toni di grigio.

1

2

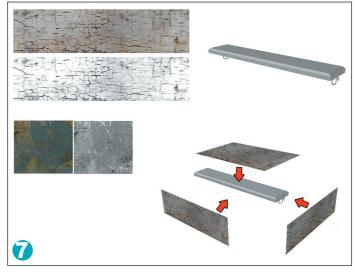
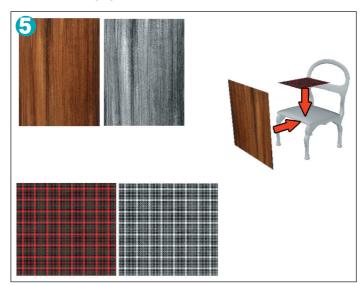
3

4-5-6 La poltroncina

Tocca adesso texturizzare la poltroncina. Anche in questo caso parliamo di texture planari in 2 diverse versioni, color e toni di grigio.

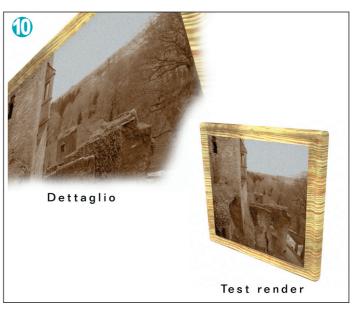
Per la poltroncina dobbiamo usare una texture di tipo legno per il corpo della stessa e una di tipo tessuto per il cuscino. In figura 4 abbiamo l'oggetto grezzo, così come da noi creato, in figura 5 possiamo vedere le texture utilizzate per mappare l'oggetto e la loro applicazione. La resa di ogni oggetto dipende strettamente dalla texture principale e dalle sue versioni in toni di grigio; in tutto questo ha un ruolo fondamentale l'illuminazione scelta per il rendering finale, cosa che analizzeremo nel proseguo. L'oggetto texturizzato completo è mostrato in figura 6.

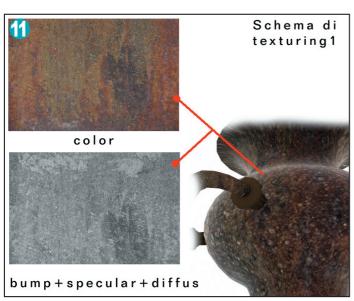
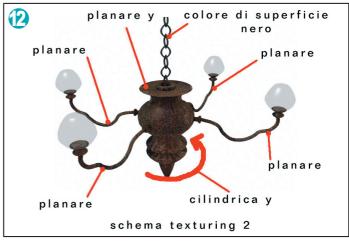
7-8 La mensola

Anche per la mensola sceglieremo delle texture di tipo legno, mentre utilizzeremo una piccola texture metallo per i sostegni della stessa. Partiamo dalla superficie della mensola, in figura 7 sono visualizzate le texture utilizzate, l'oggetto grezzo e l'asse di applicazione delle mappe. Per i sostegni in ferro utilizzeremo una "mappatura" cilindrica sull'asse y. Il risultato dell'oggetto texturizzato è visibile in figura 8. Volendo. possiamo inserire oggetti sulla mensola per aumentare il dettaglio e la resa finale dell'intera stanza in fase di rendering.

Note

Il primo resident evil a cui si è ispirato il tutorial è uscito nel lontano 1996, creando di fatto l'attuale genere "survival horror". L'intero gioco era basato su locazioni pre renderizzate in cui si muovevano personaggi poligonali 3D in realtime. Successivamente, visto il successo ottenuto, la Capcom ha prodotto diversi altri "episodi" per le varie piattaforme (l'immagine qui raffigurata si riferisce al terzo capitolo, "nemesis"). Attualmente il gioco è presente nella sua ultima versione su piattaforma GameCube di Nintendo, mantenendo inalterato lo stile tecnico e grafico di base, ma adattandosi naturalmente alle moderne tecnologie e possibilità offerte dalla macchina di casa Nintendo.

9-10 I quadri

Altro elemento da texture sono i quadri, per questi utilizzeremo una texture legno modifica per la cornice e un'immagine da inserire come tela vera e propria. In figura 9 abbiamo lo schema delle texture usate per l'oggetto. Queste vengono "mappate" in modo planare sui vari assi dell'oggetto. In figura 10 si posso "ammirare" le texture applicate all'oggetto e un test render dello stesso. Anche questo oggetto adesso è pronto per essere inserito nella scena, ma prima dobbiamo ultimare gli ultimi 2 elementi, il lampadario e gli appliques.

11-12-13 Il lampadario

Questo oggetto richiede più attenzione per il texturing. Dobbiamo suddividerlo in più superfici sulla quale andremo a texturare la stessa bitmap. In figura 11 abbiamo la texture impiegata e il corrispondente risultato in fase di rendering. Si tratta di una texture di tipo metallo con alcune modifiche per renderla tailizzabile; la stessa in seguito è stata riprodotta in toni di grigio per i canali di specularità, bump e diffusione della luce. In figura 12 possiamo vedere gli assi e la modalità di "mappatura", da utilizzare per ogni sezione del lampadario. Per il vetro dobbiamo settare il colore di base (un colore chiaro tendente al giallo pallido), una trasparenza del 10 %, diamo un fattore di luminosità al vetro stesso di circa 150% e attiviamo un effetto glow sulla sua superficie. Fatto questo, possiamo eseguire un rendering di prova dell'oggetto per controllare la corretta resa delle texture (figura 13).

Gli appliques 14

Per gli appliques utilizzeremo la stessa texture del lampadario, idem per il settario del vetro. Un esempio del rendering possiamo vederlo in figura 14. Il texturing deve essere eseguito con mappature planari per la base e la semplice superficie "vetro" per la relativa sezione.

La texture applicata

15-16-17 La scena finale

Tutto è pronto per il rendering finale, abbiamo i nostri oggetti completamente texturati da inserire in scena. Per il posizionamento delle porte e degli oggetti ogni utente ha "carta bianca". Eseguiremo il calcolo finale con un motore radiosity per aumentare il realismo, reso possibile grazie alla diffusione della luce indiretta. In figura 15 abbiamo la scena in OpenGL cosi come si presenta prima del rendering. Iniziamo con l'impostare delle luci, queste devono essere di tipo point con un "intesity falloff" attivo (diminuzione della luce in base alla distanza). Dobbiamo sistemare una point per ogni punto luce della stanza, quindi 4 per il lampadario e 1 per ogni appliques. Posizioniamo le luci poco sopra la boccia di vetro dei punti luce, questo per permettere di diffondere la luce assieme alla boccia stessa. Nel caso di stanza troppo scura abbiamo 2 modi per risolvere l'inconveniente: l'aggiunta di luci di supporto che non proiettino ombre, anch'esse con un parametro di attenuazione attivato in base alla distanza. Queste posso essere di tipo point light posizionate strategicamente in base alle zone d'ombra venutesi a creare, oppure, altra soluzione, importare il rendering in Adobe Photoshop e quindi lavorare sui parametri di "brighteness" e "contrast" sino ad ottenere il risultato che più si avvicina alle nostre aspettative. In figura 16, 17 e 18 abbiamo tre rendering eseguiti in Lightwave 7.5.

Considerazioni

Con quest'ultimo articolo abbiamo concluso la lunga sezione riguardante la creazione di una stanza 3D pre renderizzata.
I risultati del rendering possono variare, in base ad alcuni parametri, da programma a programma, quindi, non occorre cercare la totale perfezione nella riproduzione del rendering da me ottenuto. Se dovessimo ricreare altre stanze a tema potremmo riciclare alcuni dei componenti creati come le mensole o gli appliques, onde ridurre i tempi di produzione per ciò che riguarda la modellazione. Un ritocco in Adobe Photoshop può rivelarsi necessario per ritoccare parametri come luminosità e contrasto ed effettuare piccole modifiche ed aggiustamenti post rendering.
La resa finale della scena dipende fortemente dalle texture impiegate per la definizione della stessa nonché da ogni oggetto presente. Non vi resta che lanciare il rendering, ci vediamo al prossimo tutorial.

Pocket PC

Interfacciamento di applicazioni mobili con moduli di rilevazione posizionale

Moduli GPS e Pocket PC

Molte aziende hanno cominciato a sperimentare i vantaggi del rilevamento di coordinate geografiche in loco, con Pocket PC + modulo GPS, con successivo invio ad un Application Server in remoto. Esploriamo il mondo dei moduli GPS e sperimentiamo il loro interfacciamento con un Pocket PC.

File sul CD
\soft\codice
\Merge_RDA.zip

File sul Web
www.ioprogrammo.net/files/72/Merge_RDA.zip

Requisiti
HARDWARE: Pocket PC o (dispositivo Windows CE Based), Computer con almeno processore Pentium II e 128 MB di memoria, un modulo GPS per Pocket PC.

SOFTWARE: Windows 98 SE /2000/XP, IIS (oppure PWS-Personal WEB Server con Windows 9x), SQL Server 2000 con Service Pack 1 o superiore, SQL Server CE 2.0, Embedded Visual Tools.

Nel presente articolo tratteremo la progettazione di una applicazione su Pocket PC che ci permetterà di comunicare con un modulo GPS. La comunicazione permetterà, in particolar modo, di rilevare, via satellite, le informazioni di longitudine e latitudine relative alla posizione corrente. Saremo anche in grado di rilevare dal satellite informazioni di altra natura, come ad esempio l'ora esatta. Lo scopo che ci proponiamo in questo e nel prossimo articolo, è quello di progettare due applicazioni distinte ma in comunicazione tra loro:

- **L'applicazione client, su Pocket PC:** abbiamo già detto che questa applicazione permetterà di rilevare le coordinate dal satellite tramite il modulo GPS e inviarle alla parte Server;
- **L'applicazione Server:** rileva le informazioni di posizione inviate dal palmare. Le coordinate ricevute permetteranno al Server di sapere dove si trovi un operatore o un veicolo che utilizzi l'applicazione client, e di individuarlo su una mappa.

L'architettura del sistema è rappresentata in Fig. 1. Dallo schema, possiamo notare gli attori principali del sistema: il client pocket PC su cui è installata la nostra applicazione, che prende le coordinate dal satellite, e l'application Server a cui le coordinate vengono inviate. È appena il caso di precisare che possiamo inviare le coordinate al Server utilizzando diverse tecnologie di comunicazione; nell'architettura di Fig. 1 abbiamo preso in considerazione le tecnologie GPRS, GSM e LAN. Nell'applicazione supporremo di avere una connessione LAN diretta tra il palmare e il Server. In questo caso, i dati rilevati dal satellite saranno memorizzari "in locale" sul Pocket PC; una volta giunti nella rete LAN dove è installato il Server, i dati saranno sincronizzati connettendo il Pocket PC alla rete stessa. L'ipotesi di utilizzo di una connessione locale ha lo svantaggio che il palmare non potrà comunicatre in real-time la posizione del veicolo. Lo scopo del presente lavoro è solo quello di costruire una applicazione che permetta di prelevare informazioni dal GPS e trasmetterle in qualche modo ad un Application Server; ci proponiamo di trattare in un altro articolo lo studio delle metodologie con le quali attivare una connessione GPRS o GSM con il palmare, e usare tale tecnologie per rendere "veramente mobili" le nostre applicazioni.

Fig. 1: Architettura del sistema relativo alle due applicazioni del progetto.

MECCANISMO DI SCAMBIO DATI

A questo punto è importante specificare la metodologia con la quale i due attori dell'applicazione possano scambiarsi i dati. Il cuore del sistema che permetterà lo scambio dei dati sarà SQL Server CE. Sul Server installeremo SQL Server, su cui progetteremo il database che conterrà la tabella in cui memorizzeremo le coordinate corrispondenti alle diverse posizioni relative al veicolo. Il passo successivo sarà quello di effettuare la pubblicazione del Database secondo la tecnica decritta in un

precedente articolo. La pubblicazione ci consentirà di costruire una replica del database del Server sul palmare tramite la procedura di sottoscrizione. Ricordiamo che la procedura di sottoscrizione e successive sincronizzazioni dati tra Server e client Pocket PC potranno avvenire solo se sul palmare avremo installato SQL Server CE 2.0. Avvenuta la fase di sottoscrizione, la logica di funzionamento dell'applicazione palmare sarà quella di attivare un thread secondario (in background) rispetto al thread principale dell'applicazione, il quale si occuperà della rilevazione e lettura dei dati provenienti dal satellite, dalla porta seriale del palmare a cui è collegato il modulo GPS, e scrivere i dati sull'istanza locale del database. Precisiamo da subito che gli istanti in cui il thread secondario campionerà i dati dal GPS potranno essere gestiti da codice. L'applicazione sul palmare disporrà poi della logica necessaria alla sincronizzazione dei dati con il server di Database. Possiamo anticipare che lo scambio dati tra application Server e client pocket PC avviene tramite tecnologia *Merge Replication* di SQL Server. Inoltre, ogni volta che una sincronizzazione dati tra palmare e server di Database è andata a buon fine, l'application server avrà i dati aggiornati delle posizioni occupate dal veicolo in un dato periodo.

ARCHITETTURA DELLA APPLICAZIONE CLIENT

Lo scopo che ora ci proponiamo è quello di descrivere l'architettura della applicazione client sul palmare in modo da evidenziare gli oggetti necessari per il corretto funzionamento del sistema di posizionamento. In Fig. 2 è possibile notare gli attori fondamentali in gioco.

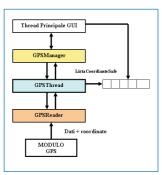



Fig. 2: **L'architettura *Client*.**

Nel CD di "ioProgrammo" potrete trovare l'applicazione completa (*GPSUtility*); noi daremo una visione d'insieme per cui l'intera logica di funzionamento possa essere ricostruita in modo semplice e senza problemi.

L'applicazione sul palmare si configura come una applicazione *multithread*; due sono i thread che lavoreranno ad un certo istante sul palmare:

- Il thread dell'applicazione principale attivato in fase di invocazione del programma.
- Il thread secondario che gestisce la comunicazione diretta con il modulo GPS e ne legge i dati ricevuti dal satellite.

L'oggetto GPSThread nello schema di Fig. 2, rappresenta proprio il thread del punto 2. Esso è attivato dal metodo Initialise dell'oggetto GPSManager. In particolare, GSPThread si serve dell' oggetto GPSReader per prelevare le coordinate (e non solo) dal GPS. Nello schema di architettura di Fig. 2, possiamo notare un Buffer che abbiamo indicato con il nome di *"Lista Coordinate Safe"*; nella corrente implementazione, questa lista simula un database in locale su cui inseriamo dei dati. La differenza sostanziale è che la lista è creata a run-time e, una volta chiusa l'applicazione, non rimarrà traccia delle posizioni rilevate. Ad ogni modo ci permetterà di fare ugualmente importanti considerazioni progettuali e di programmazione. Infatti, questa lista è utilizzata non solo dal thread GPSThread per scrivere le coordinate rilevate dal GPS, ma anche dal thread dell'applicazione principale per prelevare tali coordinate e mostrarle all'utente. Ora stiamo per capire il motivo per cui la lista di coordinate è stata definita "safe": dobbiamo garantire un accesso sicuro e in mutua esclusione sul Buffer. Chiunque di voi abbia avuto a che fare con le applicazioni relative alla "programmazione concorrente" avrà subito capito che ci troviamo di fronte al classico problema dei "Lettori e degli Scrittori": in particolare dobbiamo sincronizzare i thread in modo tale che quando si inizia una operazione di lettura, non possa essere eseguita una operazione di scrittura e viceversa. Il Problema viene risolto con l'ausilio dei "semafori" che permettono di disciplinare l'azione di più thread su una risorsa condivisa. Questi oggetti sono presenti anche in Windows CE così come la gestione dei thread.

APPLICAZIONE GPSUTILITY

L'applicazione GPSUtility si configura come un'applicazione In Embedded Visual C++ 3.0 Dialog Based. In Fig. 3 è rappresentata le maschera principale dell'applicazione. In Fig. 4 abbiamo evidenziato, con un'ellisse nel Project Explorer, la classe CGPSUtilityDlg associata alla risorsa Dialog di Fig. 4. In quest'ultima classe possiamo notare la presenza della variabile membro privata *m_gpsManager*, istanza della classe *CGPSManager*. Questa classe fornisce i metodi atraverso i quali inizia il processo di lettura delle coordinate rilevate dal satellite. Nella maschera di Fig. 3 possiamo notare i seguenti tasti:

Fig. 3: **La maschera con le diverse funzioni.**

- **Start GPS**: viene fatto partire il thread secondario per la lettura delle coordinate dal modulo GPS. La funzione di gestione dell'evento *click* è molto semplice:

```
m_gpsManager.Initialise();
```

Pocket PC

Application Server

Nel prossimo articolo, in cui ci occuperemo anche dell'Application Server e della interazione con un Pocket PC, aggiungeremo all'applicazione client la classe di accesso ai dati e di sincronizzazione.
Non l'abbiamo fatto ora solo perchè ci siamo voluti concentrare sugli aspetti dell'architettura necessari per l'interfacciamento con un qualunque modulo GPS. Il primo oggetto che prendiamo in considerazione è il GPSManager. Tale oggetto si configura come quello che gestisce l'inizio e la fine del processo di comunicazione con il modulo GPS. A tal fine, *GSPManager* dispone dei metodi *Initialise* e *Closedown*.

Pocket PC

Coordinate

Le coordinate rilevate dal GPS possono essere facilmente lette progettando una applicazione per Pocket PC. La rilevazione continua delle coordinate permette di memorizzarle e inviarle ad una application server per consentire in ogni momento la conoscenza della posizione di un veicolo o di un operatore e per diversi scopi. Uno degli scopi potrebbe essere quello rilevare la posizione attuale di diverse ambulanze e, nel caso di un soccorso urgente, contattare direttamente quella più vicina al luogo di interesse.

GPS

GPS *(Global Positioning System)* rappresenta il sistema con il quale possiamo rilevare, con una accurattezza di circa 100m, con un ricevitore adeguato le coordinate dal sistema di satelliti in orbita. Anche i Pocket PC possono essere utilizzati per connettersi ad un modulo GPS esterno sia attraverso il connettore seriale che tramite altra interfaccia.

- **Coordinate Correnti**: vengono mostrate in una finestra di dialogo le ultime coordiante rilevate. L'implementazione del metodo di gestione dell'evento *click* è la seguente:

```
CCoordinate cc;
  if(m_gpsManager.GetCoordinate(cc)) {
      AfxMessageBox(_T("Latitudine = ") +
                              cc.GetLatitudine());
     AfxMessageBox(_T("Longitudine = ") +
                              cc.GetLongitudine());
     AfxMessageBox(_T("Ora = ") + cc.GetOra());}
  else {
     AfxMessageBox(_T("Non ci sono coordinate al
                                   momento!")); }
```

Fig. 4: **Oggetti della classe *CGPSUtilityDlg.***

Possiamo notare la dichiarazione di un oggetto di tipo Coordinate. Questa classe è molto semplice e permette di modellare l'informazione rilevata dal satellite: le sue proprietà saranno valorizzate all'atto della chiamata del metodo GetCoordinate(cc) sull'istanza *m_gpsManager*; rimandiamo al codice sul CD per la struttura della classe Coordinate.

- **Stop GPS**: termina il thread di rilevazione delle coordinate dal satellite. Il metodo associato al click è il seguente: m_gpsManager.CloseDown();

CLASSI GPSMANAGER E GPSTHREAD

In Fig. 5, possiamo notare la struttura di questa classe. In particolare, possiamo notare la presenza di due oggetti come variabili membro di classe:

GPSThread m_gpsThread: rappresenta il thread secondario, l'esecuzione del quale fa iniziare il processo di lettura delle coordinate dal GPS se esso è collegato e alimentato;

ListaCoordinateSafe m_listaCoordinateSafe: rappresenta l'implementazione del contenitore delle informazioni rilevate dal thread e gestita in maniera safe.

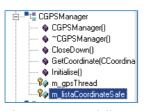



Fig. 5: **Struttura della classe *CGPSManager.***

Nel progetto sviluppato abbiamo costruito in Windows CE lo schema di principio di un Thread; abbiamo chiamato questa classe *CThread*. La classe CThread rappresenta la classe base da cui derivare tutti i nostri thread che effettuano operazioni specifiche. Per forzare questo comportamento abbiamo dichiarato dei metodi virtuali. Ecco come si presenta il file di intestazione della classe:

```
class CThread {
public:
  CThread();
  virtual ~CThread();
// Public interface
  virtual bool Start();
  virtual bool Stop();
  virtual DWORD Suspend();
  virtual DWORD Resume();
protected:
  virtual int Run() = 0;
// Implementation
protected:
  void Lock()   const;
  void Unlock() const;
// procedura del thread di classe
protected:
  static DWORD WINAPI ThreadProc(LPVOID lpParameter);
//Procedura del thread istanza
protected:
  DWORD ThreadProc();
protected:
  HANDLE          m_hThread;
  bool            m_fEndThread;
mutable CRITICAL_SECTION m_cs; };
```

La classe CThread possiede un membro *m_hThread*, che rappresenta un handle al processo, che verrà effettivamente creato nel costruttore della classe tramite un'apposita API del sistema operativo Windows CE. Per garantire l'accesso sicuro alle proprietà del thread è necessario usare un semaforo rappresentato dall'oggetto *m_cs* di tipo *CRITICAL_SECTION*. Gli oggetti di tipo *CRITICAL_SECTION* permettono di costruire un protocollo sicuro per l'accesso in modo sincronizzato a parti di codice che sono condivise da più processi.
Prima di poter accedere a qualunque risorsa condivisa, è necessario chiedere il permesso per entrare in sezione critica e bloccarsi fino a quando non ne ottengo il diritto. Dopo aver ottenuto l'accesso nella alla sezione critica, e aver utulizzato le risorse condivise, è necessario rilasciare l'uso della la risorsa per non bloccare eventuali altri processi che ne richiedessero accesso. Questo protocollo di "buone maniere" è ottenuto con i metodi *Lock* e *UnLock* di cui forniamo di seguito l'implementazione:

```
void CThread::Lock() const
{ ::EnterCriticalSection(&m_cs); }
void CThread::Unlock() const
{ ::LeaveCriticalSection(&m_cs); }
```

Dal codice notiamo che si utilizzano le funzioni globali *EnterCriticalSection* e *LeaveCriticalSection* per ottenere

questo risultato, passando per riferimento l'oggetto *m_cs*. Tutte le funzioni relative alla classe *Thread* inizieranno con *Lock* e finiranno con *UnLock*, proprio per rispettare questo protocollo di sincronizzazione tra processi. Vediamo ora il costruttore della classe:

```
CThread::CThread() {
  ::InitializeCriticalSection(&m_cs);
  m_hThread     = NULL;
  m_fEndThread = false; }
```

Come possimo notare dal codice, nella lista viene inizializzato l'oggetto della sezione critica. L'handle del thread viene posta a *NULL* e la variabile *m_fEndThread*, che verifica che, se il thread è in esecuzione, è posta a *false*. La effettiva creazione dell'istanza del thread e la sua esecuzione viene attivata con la funzione *start()* seguente:

```
bool CThread::Start() {
  Lock();
  m_fEndThread = false;
  if (m_hThread == NULL) {
    DWORD dwThreadId;
    m_hThread = ::CreateThread(NULL, 0, ThreadProc,
                                   this, 0, &dwThreadId); }
  Unlock();
  return (m_hThread != NULL); }
```

Con l'ausilio della funzione *CreateThread* possiamo costruire l'istanza del thread; alla funzione passiamo anche il puntatore alla funzione che dovrà essere eseguita ripetutamente dal thread fino a quando non sarà bloccato con la funzione *Stop*. In ultima analisi il progetto della classe è stato fatto in modo tale che la funzione *ThreadProc* richiamerà a sua volta l'implementazione del metodo *Run()*. Quando dovremo costruire thread personalizzati, sarà dunque necessario preoccuparci di implementare il metodo *Run()*. Alla luce delle considerazioni sulla classe *CThread*, costruiamo la classe *CGPSThread* che la estende.
La classe utilizza un oggetto della classe *m_GpsReader* per aprire la porta di comunicazione e interfacciarsi con il moduolo GPS. Nella funzione di *Start()* possiamo notare che viene aperta la porta seriale di connessione al GPS; una volta che tale operazione è andata a buon fine, possiamo chiamare il metodo *start()* sul thread in modo che possa iniziare il processo di rilevazione della coordinate. Con il metodo *Stop* chiudiamo la porta di connessione al GPS e blocchiamo l'esecuzione del thread. Vediamo di seguito l'implementazione del metodo *Run()*:

```
int GPSThread::Run() {
  CCoordinate cc;
  m_GpsReader.ReadCordinates(cc);
  ::Sleep(1000);
  if(m_listaCoordinateSafe->InsertPosition(
                 cc.GetLatitudine(),cc.GetLongitudine())) {
    TRACE(_T("COORDINATA DELLE ORE ")+
                          cc.GetOra() +_T("\n")); }
  return 1; }
```

La logica è molto semplice: leggiamo le coordinate, invocando il metodo *ReadCoordinates* sull'oggetto *m_Gpsreader*, e le scarichiamo in modo safe sulla lista delle coordinate rilevate. Inoltre, se l'operazione di inserimeto nella struttura dati è andata a buon fine, facciamo un trace nella finestra di Debug delle informazioni rilevate. L'operazione di *TRACE* sarà effettiva solo se siamo nella modalità debug della applicazione.

CLASSE GPSREADER

Dallo schema di architettura di Fig. 2, abbiamo visto come questo modulo comunichi direttamente con il GPS. Vediamo in dettaglio la struttura delle informazioni che vengono lette dal modulo. Il Modulo GPS su cui abbiamo effettuato le nostre prove è Destinator. Destinator rappresenta un modulo GPS esterno che si collega alla porta seriale del Pocket PC.
Il Pocket PC di riferimento è un iPAQ 3970 della Compaq. Per quanto riguarda la lettura dei dati da un modulo GPS; diciamo subito che questa operazione è ottenuta utilizzando le funzioni standard della comunicazione seriale.
Nel file di esempio contenuto nel CD-Rom allegato, vediamo l'implementazione della operazione di apertura della porta seriale del GPS ottenuta tramite la funzione *OpenPort* della classe *GPSReader*.
La funzione *OpenPort* determina prima di tutto il nome della porta a cui connettersi (tipicamente è la COM1 per un GPS esterno come il nostro). È stata utilizzata la funzione *CreateFile* per aprire la porta; inoltre per poter configurare la stessa è stata utilizzata la struttura DCB. Una variabile di tipo *DCB* rappresenta una struttura che definisce i valori da impostare per controlla la porta seriale cui è collegato un dispositivo con cui dialogare. Seguire attentamente i valori con cui abbiamo configurato la porta seriale, poiché è di vitale importanza per il corretto funzionamento dell'operazione successiva: la lettura dei segnali provenienti dal satellite.
Non vogliamo appesantire la trattazione, per cui non andremo in dettaglio nella comprensione dei parametri di configurazione della porta. Diciamo solo che se l'operazione di apertura della porta è andata a buon fine, verrà restituito un valore booleano true. Una volta che la porta seriale è stata aperta e opportunamente configurata, è possibile leggere il segnale proveniente dal satellite. I dispositivi GPS producono informazioni posizionali in accordo allo standard NMEA 0183. Lo standard NMEA è basato su linee di comando testuali. Ogni linea dati di comando viene definita sentenza. Inoltre, le sentenze sono mandate in output dal GPS in maniera continua. Da questo si capisce l'importanza di un thread per la cattura di tali informazioni. Lo standard, inoltre, stabilisce che la velocità di trasmissione

RMC

La sentenza di tipo RMC rappresenta, una particolare linea di caratteri che descrivono dati rilevati dal satellite.

Pocket PC

Quale GPS?

Il Modulo GPS su cui sono state effettuate le nostre prove è il Destinator. Questo GPS viene fornito unitamente al software PowerLoc e a delle mappe geografiche che consentono di riprodurre del tutto le funzionalità dei moduli GPS installati nelle auto. In Fig. 7

possiamo vedere la maschera di caricamento iniziale del programma. Nelle Fig. 8 e 9 invece si può notare la posizione rilevata e visualizzata sulla mappa installata sul Pocket PC. Si può acquistare dalla casa produttrice un SDK con il quale interfacciarsi con l'applicazione a corredo del Destinator.

L'incoveniente dell'applicazione PowerLoc è che essa apre la porta seriale per leggere le coordinate dal GPS, per cui una nostra applicazione non potrebbe aprirla dato che solo una per volta può farlo.

dei comandi sia di 4800 Baud, ma per molti GPS possono essere selezionate diverse velocità di trasmissione. Un esempio tipico di sentenza è la seguente:

```
$GPRMC,195531,A,5326.986,N,00610.147,W,000.0,360.0,
                                  170500,007.2,W*7F
```

Quella appena scritta, rappresenta una tra le possibili sentenze in uscita da un modulo GPS. In particolare, abbiamo qui rappresentato la sentenza RMC. La sentenza di tipo RMC rappresenta, quindi, una particolare linea di caratteri che descrivono dati rilevati dal satellite. In particolare, ogni sentenza inizia con il carattere $; le due lettere seguenti rappresentano l'identificativo del tipo di dispositivo, in questo caso "GP" indica che il GPS di riferimento è di tipo Garmin. I tre caratteri successivi rappresentano il tipo di comando, in questo caso RMC. I dati nella sentenza sono separati da virgole.
Ogni sentenza termina con un carattere di "*" seguito da un numero che rappresenta la checksum per la verifica della correttezza della sequenza di caratteri. Il primo dato contenuto nella sentenza di esempio rappresenta l'ora in cui la posizione è stata rilevata dal satellite nella notazione "hhmmss", ovvero in questo caso le 19:55:31. Il carattere seguente l'ora di rilevazione della posizione serve per sapere se è buono lo stato di rilevazione del segnale dal satellite. Sono possibili due valori:

- **A**: il segnale è buono.
- **V**: valore di warning ovvero la rilevazione potrebbe essere sospetta.

Il dato successivo 5326.986 rappresenta la latitudine corrente espressa in gradi nel range 0-90 gradi, minuti, e frazioni di minuto; in questo caso la latitudine è 53 gradi, 26 minuti e 1/986 di minuto. La longitudine corrente, espressa nello stesso formato della latitudine, è 00610.147. Infine, 170500 è la data di rilevazione delle coordinate nel formato "ggmmaa", che corrisponde al 17 Maggio del 2000. Un altro tipo di sentenza simile alla precedente, ma senza l'indicazione della data del giorno, è la seguente (corrispondente al comando *GGA*):

```
$GPGGA,195531,5326.986,N,00610.147,W,1,07,1.5,24.7
                                       M,54.0,M,,*6°
```

Rimandiamo alle specifiche dello standard NMEA per una esaustiva rassegna delle tipologie di comandi in output da un sistema GPS e il significato dei dati contenuti in ogni sentenza di comando.

METODO DI LETTURA DEI DATI

A questo punto analiziamo il metodo *ReadCoordinates* della classe *GPSReader*, con la quale leggiamo le sentenze provenienti dalla porta seriale appena aperta.
La logica della funzione è quella tipica di lettura da una porta seriale. Innanzi tutto, viene invocata la funzione *SetCommMask* passando il riferimento allo *HANDLE* della porta COM, e specificando alcuni valori costanti in OR che rappresentano gli eventi che bisogna gestire. La registrazione di tali eventi viene fatta con il tramite della funzione di Windows CE *WaitCommEvent*. In particolare, specificare la costante *EV_RXCHAR* nel metodo *SetCommMask* vuol dire voler rilevare che un carattere è stato ricevuto sulla porta seriale e posto nel buffer.
Dopo l'impostazione degli eventi, inizia il ciclo di lettura dalla porta seriale tramite la funzione *ReadFile* e leggendo un Byte alla volta. Durante il ciclo di lettura viene ricostruita la sentenza che ci interessa. Nel caso specifico siamo interessati sia alla sentenza di tipo RMC che sia GGA, e abbiamo scritto due funzioni di parsing che prelevano le informazioni: le funzioni in questione sono *ParseRMC* e *ParseGGA* che vi invitiamo a visionare nel CD di "ioProgramo". Le funzioni in questione permetteranno di valorizzare gli appositi campi della classe di tipo Coordinate che passiamo come parametro.

Fig. 6: Valori di Latitudine, Longitudine e Tempo delle coordinate correnti.

Per verficare che tutto funziona bene, mandiamo in esecuzione la nostra applicazione. Dopo aver aperto la porta seriale con il tasto "Start GPS", premere il tasto "Coordinate Correnti"; verranno visulizzate in sequenza tre finestre di messaggi con i valori della Latitudine (Fig. 6-A), della Longitudine (Fig. 6-B) e dell'ora (Fig. 6-C) in cui le coordinate sono state rilevate dal satellite.

CONCLUSIONI

In questo primo articolo sulle applicazioni di posizionamento per Pocket PC, abbiamo descritto una applicazione con cui riusciamo a catturare i dati da un sistema GPS.
Nel prossimo articolo prenderemo le mosse da questa applicazione palmare per aggiungervi le funzionalità necessarie per l'accesso ai dati con SQL Server CE, e la logica di sincronizzazione.
Inoltre, svilupperemo la parte Server in APS .NET con la quale potremo usare le coordinate rilevate dal satellite per visualizzarle su una mappa. Per la logica di disegno su mappe geografiche ci serviremo di *MapPoint.NET*.

Ing. Elmiro Tavolaro

Codice di qualità

Tecniche per la stesura di codice manutenibile (parte seconda)

La qualità del codice Java

In questo appuntamento illustreremo le linee guida per realizzare codice ben scritto e mostreremo uno strumento per la verifica della qualità del nostro codice.

Nell'articolo precedente abbiamo analizzato le problematiche di qualità legate al concetto di sviluppo del software, distinguendo la qualità semantica, legata a come il software viene progettato ed organizzato, da quella sintattica, più indirizzata invece a come questo viene scritto. Concentrandoci sul secondo aspetto, abbiamo cercato di capire quali regole sintattiche debbano essere rispettate e quali metriche si debbano utilizzare per la verifica di conformità del nostro codice con le regole stesse. In particolare, dal punto di vista delle regole da seguire per l'analisi della qualità sintattica del codice Java, ci siamo basati sulle *Code Conventions for the Java Programming Language* fornite dalla stessa Sun Microsystem. In questo secondo articolo analizzeremo le ultime regole indicate nelle *Code Conventions* ed utilizzeremo un tool open source, disponibile anche come plugin per Eclipse, in grado di analizzare il codice di un nostro progetto e di dirci dove questo non è conforme al nostro pacchetto personalizzato di regole.

ELEMENTI DI SEPARAZIONE

Uno dei requisiti richiesti al codice di qualità è la sua leggibilità anche, e soprattutto, per chi non lo ha scritto e non lo conosce. La sua impaginazione è quindi importante e di conseguenza diventa essenziale sapere come gestire gli elementi di separazione del codice, cioè spazi bianchi e linee vuote.

Le linee vuote

In generale aumentano la leggibilità facendo risaltare sezioni di codice che sono tra loro logicamente correlate. Una doppia linea bianca dovrebbe essere sempre usata per separare sezioni omogenee di codice in un file sorgente e fra le definizioni di classe e di interfaccia. In altre situazioni nelle quali sia necessario separare porzioni di codice tra loro diverse può essere sufficiente un'unica linea vuota, come nelle seguenti:

- Fra metodi della stessa classe
- Fra le dichiarazione delle variabili locali in un metodo e la sua prima istruzione
- Prima di un commento di blocco o a singola linea
- Fra sezioni logiche all'interno di un metodo per migliorarne la leggibilità

Gli spazi bianchi

Dopo l'impaginazione verticale anche quella orizzontale vuole la sua parte, ecco quindi la necessità di utilizzare gli spazi bianchi in molte situazioni, vediamole:

- Quando c'è una parola chiave seguita da una parentesi:

```
while (true) {
    ... }
```

- Dopo le virgole che separano gli argomenti di un metodo:

```
Database dbProp = (Database)rwp.getPropObj(
                                    "Database", nome);
```

- Tutti gli operatori binari eccetto il punto dovrebbero essere separati dai loro operandi tramite spazi. Viceversa gli spazi bianchi non dovrebbero mai separare gli operatori unari come l'operatore meno, l'incremento *("++")*, e il decremento *("--")* dai loro operandi.

```
a += c + d;
a = (a + b) / (c * d);
while (d++ = s++) {
   n++;}
printSize("size is " + foo + "\n");
```

- Le espressioni in un'istruzione *for* dovrebbero essere separate da spazi bianchi:

Checkstyle

Checkstyle è uno strumento open source di valutazione automatica della conformità del codice Java che applica un set configurabile e personalizzabile di regole. L'utilizzo tradizionale di Checkstyle prevede un insieme di tool linea di comando che permettono di selezionare quale codice valutare ed attraverso quale configurazione di regole. Le regole, e le loro severity, sono configurate su file XML. Per abbassare il tempo di apprendimento e per renderlo uno strumento integrabile, esiste una versione di Checkstyle distribuita sotto forma di plugin per Eclipse.

\soft\checkstyle-3.1.zip

Codice di qualità

Data Hiding

Il Data Hiding è uno dei fondamenti del paradigma Object Oriented e consiste nel non avere, all'interno di una classe, oggetti e variabili pubblici che possano essere letti e scritti direttamente senza l'utilizzo di appositi metodi di lettura e scrittura.

Convenzioni

Le *Code Conventions for the Java Programming Language* sono delle linee guida sintattiche promosse direttamente da Sun Microsystem e che hanno l'obiettivo di indicare come deve essere scritto il codice Java per poter essere immediatamente comprensibile anche da chi non lo ha direttamente scritto.

```
for (expr1; expr2; expr3)
```

- *I cast*, per la loro importanza, dovrebbero essere seguiti da uno spazio bianco:

```
myMethod((byte) aNum, (Object) x);
myMethod((int) (cp + 5), ((int) (i + 3)) + 1);
```

Naming

Le convenzioni di *naming* permettono di avere software più comprensibile e leggibile anche per chi non lo ha scritto. In generale esiste una regola per ogni tipo di identificatore, vediamo le principali:

- **I package:** il prefisso univoco deve essere scritto in minuscolo e rappresentare, per quanto possibile, un suffisso di dominio di massimo livello *(com, net, org).*
- **Le classi**: il nome di una classe e delle interfacce sono sostantivi con l'iniziale (di ogni parola interna costituente il nome completo) maiuscola.
- **I metodi:** il nome di un metodo è un predicato con l'iniziale minuscola, ma con le iniziali di eventuali parole interne scritte invece in carattere maiuscolo.
- **Gli oggetti:** il nome di un'istanza di classe è un sostantivo che lo descrive funzionalmente, ha l'iniziale minuscola, ma le iniziali di eventuali parole interne sono scritte in carattere maiuscolo. E' permesso l'utilizzo di nomi dotati di una singola lettera per variabili particolari come gli indici dei cicli.
- **Le costanti:** sono scritte in caratteri maiuscoli e la separazione tra una parola e l'altra è data da un carattere underscore.

CONSUETUDINI DI SCRITTURA DEL CODICE

Quando si scrive software è bene adattarsi alle regole di stesura del codice proprie del gruppo di lavoro con il quale si collabora. In ogni caso esistono alcune convenzioni e consuetudini che dovrebbero sempre essere seguite per ottenere software di qualità sintattica migliore.

Metodi di classi statiche

In presenza di classi statiche è buona norma riferirsi ai loro metodi utilizzando la sintassi *AClass.classMethod();* e non invece la sintassi propria dei metodi tradizionali *anObject.classMethod();*

Assegnamento di valori alle variabili

La prima cosa da evitare è l'utilizzo di una sola istruzione per assegnare lo stesso valore a variabili differenti attraverso la sintassi *fooBar.fChar = barFoo.lchar = 'c';* che è davvero difficile da leggere. Altra cosa da non fare è la ricerca dell'ottimizzazione stretta nel codice, magari attraverso l'utilizzo di assegnamenti uno dentro l'altro, come in questo caso:

d = (a = b + c) + r;

Lo stesso codice può (e deve) essere scritto con una sintassi più comprensibile come questa:

a = b + c;
d = a + r;

Parentesi

La lettura del codice deve molto, come si diceva, al raggruppamento di oggetti omogenei ed alla separazione di questi gruppi da altri che tra loro non lo siano. Per questi motivi le parentesi possono essere usate per raggruppare tra loro espressioni che debbano in qualche modo essere separate da altre, non solo per la corretta esecuzione del codice, ma anche per motivi di leggibilità. Il codice della riga seguente, per esempio:

```
if (a == b && c == d)
```

è da evitare perché potrebbe causare confusione in lettura da parte di chi non abbia estremamente chiare le problematiche di precedenza tra gli operatori. In ogni caso la stessa riga di codice può essere scritta in questo modo:

```
if ((a == b) && (c == d))
```

risultando più leggibile e comprensibile.

I VALORI DI RITORNO

Il codice dovrebbe essere scritto il più semplicemente possibile, nel caso del valore ritornato da un metodo è possibile scrivere una frammento come questo:

```
if (booleanExpression) {
  return true; } else {
  return false; }
```

E' chiaro però che l'obiettivo è restituire al chiamante lo stesso valore dell'espressione che viene valutata, pertanto questo codice andrebbe riscritto in questo modo:

```
return booleanExpression;
```

Allo stesso modo è importante conoscere a fondo tutti i costrutti del linguaggio, anche quelli che si utilizzano di meno, in quanto spesso ci offrono grandi potenzialità.
Prendiamo in esame, per esempio, questo frammento di codice:

```
if (condition) {
  return x; }
return y;
```

Dovremmo sapere che lo stesso risultato si può ottenere attraverso questo operatore ternario:

Codice di qualità

```
return (condition ? x : y);
```

Proprio con l'utilizzo di questo operatore è necessario porre la massima attenzione alla condizione che, nel caso in cui sia una condizione composta, deve essere racchiusa tra parentesi in questo modo:

```
return ((x > y) ? x : y);
```

IL CODICE PERFETTO?

Esiste un frammento di codice che soddisfi tutte le norme indicate nelle *Code Conventions for the Java Programming Language?* La risposta di chi sviluppa codice Java può essere di due tipi:

- Tutto il mio codice è conforme alle *Code Conventions!*
- Nel mio codice conta il risultato e non la forma!

Il buon senso, come sempre, deve avere la meglio sugli estremismi da una parte e dall'altra, quindi è importante che il software soddisfi i requisiti per cui è scritto, ma è altrettanto importante che sia scritto in maniera comprensibile per evitare che sia troppo oneroso un intervento di manutenzione. In ogni caso c'è da dire che il codice perfetto, almeno dal punto di vista formale, esiste ed è quello indicato come esempio dalle stesse *Code Conventions*, vediamolo e cerchiamo di capire quanto assomigli al nostro.

```
/*
 * @(#)Blah.java        1.82 99/03/18
 *
 * Copyright (c) 1994-1999 Sun Microsystems, Inc.
 * 901 San Antonio Road, Palo Alto, California, 94303, U.S.A.
 * All rights reserved.
 *
 * This software is the confidential and proprietary information of Sun
 * Microsystems, Inc. ("Confidential Information"). You shall not
 * disclose such Confidential Information and shall use it only in
 * accordance with the terms of the license agreement you entered into
 * with Sun.
 */
package java.blah;
import java.blah.blahdy.BlahBlah;
/**
 * Class description goes here.
 *
 * @version        1.82 18 Mar 1999
 * @author         Firstname Lastname
 */
public class Blah extends SomeClass {
    /* A class implementation comment can go here. */
    /** classVar1 documentation comment */
    public static int classVar1;
    /**
     * classVar2 documentation comment that happens to be
     * more than one line long
     */
    private static Object classVar2;
    /** instanceVar1 documentation comment */
    public Object instanceVar1;
    /** instanceVar2 documentation comment */
    protected int instanceVar2;
    /** instanceVar3 documentation comment */
    private Object[] instanceVar3;
    /**
     * ...constructor Blah documentation comment...
     */
    public Blah() {
        // ...implementation goes here... }
    /**
     * ...method doSomething documentation comment...
     */
    public void doSomething() {
        // ...implementation goes here... }
    /**
     * ...method doSomethingElse documentation comment...
     * @param someParam description
     */
    public void doSomethingElse(Object someParam) {
        // ...implementation goes here...  } }
```

STRUMENTI DI VALUTAZIONE

Una delle cose da evitare in generale, quando si scrive un frammento di codice, è l'attenzione maniacale alla forma, proprio perché, come è stato detto in precedenza, il buon senso deve avere la meglio comunque sugli estremismi di parte. Poiché chi scrive il codice si concentra soprattutto sull'implementazione dell'algoritmo, è bene produrre, tanto per iniziare, uno scheletro funzionante del pezzo di codice in stesura ed in un secondo momento si potrà rivedere la forma analizzando nuovamente il codice e conformandolo alle specifiche di qualità formale esistenti, qualunque esse siano.
Questa seconda parte, però, spesso viene trascurata perché si considera un'attività secondaria che in ogni caso non pregiudica il funzionamento del codice e soprattutto davvero noiosa e quasi impossibile da compiere senza errori. Perché allora non appoggiarci a strumenti di valutazione automatica della conformità del nostro codice con un set di regole configurabili, magari open source? Di strumenti di questo genere ne esistono molti, ma si tratta spesso di oggetti commerciali molto sofisticati e costosi, ragion per cui spesso non si ha la possibilità di utilizzarli e di apprezzarne i vantaggi.
In questo panorama un po' desolante spicca però la presenza di un tool open source che soddisfa ampiamente i nostri requisiti: si tratta del progetto *Checkstyle*, giunto alla versione 3.1 e disponibile gratuitamente su

Vantaggi

L'utilizzo di convenzioni, in generale, è una buona abitudine che permette a chi lavora in gruppo di abbassare i problemi di comprensione del codice scritto da altri. Altro vantaggio di un approccio rigido alle convenzioni è la possibilità di manutenere con più facilità il codice scritto anche a distanza di molto tempo. Tutto questo si traduce in una implicita qualità sintattica del codice prodotto.

Codice di qualità

sourceforge all'indirizzo *http://checkstyle.sourceforge.net/*.

CHECKSTYLE

Una volta scaricato il software e scompattato l'archivio ci troveremo di fronte all'alberatura visibile Fig. 1.

Fig. 1: Alberatura completa che si ottiene dopo avere decompresso l'archivio di distribuzione di CheckStyle.

Checkstyle è principalmente un tool a linea di comando che esegue un controllo di conformità su un file Java (o un gruppo di file fino ad un intero progetto grazie anche alla completa integrazione con *ANT*). Le regole da verificare ed il grado di severity di una eventuale non conformità sono configurabili e memorizzabili in file XML per i quali è presente un'apposita grammatica. Insieme al pacchetto sono disponibili già, come esempio, due file di configurazione pronti per l'uso: uno contenente un pacchetto di regole personalizzato di *Checkstyle* ed un altro contenente una completa rimappatura delle *Code Conventions* di cui abbiamo parlato in precedenza. Proviamo ad analizzare qualche pezzo di codice Java utilizzando il tool a linea di comando.
Per prima cosa è necessario includere nel classpath tutti gli archivi *JAR* presenti nel pacchetto di *Checkstyle*, quindi siamo pronti ad analizzare un file Java. Prendiamo come cavia il sorgente della classe *ListTag* che implementa una delle tag library dell'applicazione multicanale presentata nel numero di ioProgrammo del mese scorso. Vogliamo verificare il file secondo le *Code Conventions*, quindi utilizzeremo il file *sun_checks.xml* messo a disposizione da *Checkstyle*. Il comando che dovremo eseguire per eseguire questo controllo è il seguente:

```
java com.puppycrawl.tools.checkstyle.Main \
-c percorso_di_checkstyle/docs/sun_checks.xml \
percorso_del_sorgente\ListTag.java
```

ed il risultato di questo test è visibile in Fig. 2. Si tratta, come si può notare, di una lunga sequenza di warning, a testimonianza del fatto che il codice, seppur funzionante, non è dotato di grande qualità sintattica. Certo, analizzare questa sequenza di warning e tentare di porre rimedio manualmente ad ogni singola segnalazione potrebbe essere un lavoro lungo e noioso, proprio quello che si deve evitare se vogliamo essere invogliati a rispettare le regole sintattiche. Ecco quindi che ci viene incontro un altro strumento, che fa da ponte tra *Checkstyle* ed *Eclipse*, (ambiente di sviluppo open source che potete trovare nel CD allegato a ioProgrammo n.71, in edicola il mese scorso).

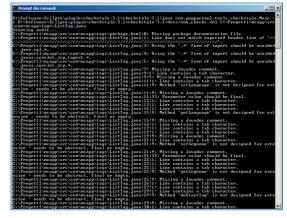



Fig. 2: La lunga sequenza di warning ottenuta dall'esecuzione di un test con CheckStyle su un file sorgente.

CHECKSTYLE COME PLUGIN PER ECLIPSE

Lo strumento di cui parliamo è un plugin per Eclipse con lo stesso motore di *Checkstyle*, progetto open source anch'esso ospitato su sourceforge scaricabile direttamente dal sito *http://eclipse-cs.sourceforge.net/*.

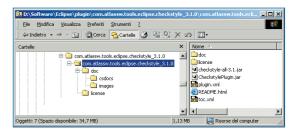



Fig. 3: Alberatura completa che si ottiene dopo aver decompresso l'archivio di distribuzione del plugin per Eclipse.

Una volta scaricato il software e scompattato l'archivio ci troveremo di fronte all'alberatura visibile in Fig. 3, per l'installazione non dovremo fare altro che copiare la cartella *com.atlassw.tools.eclipse.checkstyle_3.1.0* all'interno della directory plugins di Ecplise. La configurazione del plugin è un'operazione molto importante, in

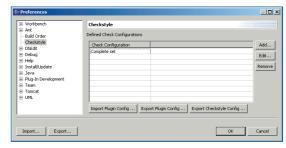



Fig. 4: Interfaccia di definizione delle configurazioni del plugin in Eclipse.

La qualità del codice

**L'analisi qualitativa di un frammento di codice si divide in due grandi rami: l'analisi della qualità sintattica e l'analisi della qualità semantica.
La qualità sintattica, e quindi la conformità del codice con certe regole e convenzioni, può essere effettuata anche con strumenti automatici di validazione. La qualità semantica invece, cioè l'utilizzo intelligente dei costrutti di programmazione in funzione dell'algoritmo da implementare, non è sempre verificabile in maniera automatica se non per approssimazioni, pertanto è bene comprenderne le problematiche per produrre codice robusto e qualitativamente efficace.**

Fig. 5: Ogni singola regola di conformità è configurabile nei minimi dettagli.

quanto ci permette di selezionare quali regole applicare ed in quali contesti. Questa viene fatta dalla sezione *window/preferences* attraverso l'interfaccia di Fig. 4, nella quale è possibile definire diverse configurazioni, ciascuna delle quali contiene tutte e sole le regole che saranno applicate e per ogni regola è possibile determinare se la non conformità con questa sia da rilevare come informazione, warning o errore. In Fig. 5 è visibile la configurazione puntuale di una singola regola. Agendo in questo modo è possibile ottenere set di regole personalizzate a livello di azienda o di singolo progetto per eliminare regole che non si ritengono importanti, oppure obbligare gli sviluppatori a rispettarne alcune che al contrario si reputano fondamentali per la qualità della produzione. Una volta configurato il set di regole da applicare, possiamo selezionarlo attraverso l'interfaccia di Fig. 6. A questo punto, confermando

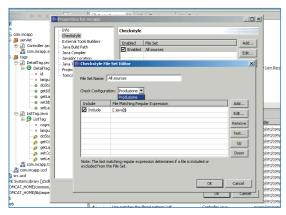



Fig. 6: Selezione del set di regole da applicare per eseguire la verifica di conformità.

le selezioni, avviene l'esecuzione del test che utilizza il motore di *Checkstyle*, attraverso l'applicazione del set di regole che abbiamo confezionato su misura per il nostro progetto. Il risultato dell'esecuzione del test, visibile in Fig. 7, è l'inserimento nella list dei task da eseguire di Eclipse, di un task per ogni violazione alle regole che il motore di *Checkstyle* ha rilevato. Questo comportamento fa si che, selezionando la singola segnalazione sia possibile posizionarsi direttamente con l'editor nel file che corretto e nella posizione esatta dove è stata rilevata la non conformità. A questo punto la gestione della qualità sintattica diventa un processo di routine che coinvolge tutti gli sviluppatori, ciascuno per le proprie porzioni di codice e non è più un'attività separata che porta via tempo. In questo modo si ottimizzano gli sforzi di correzione del codice, consentendo a chi sviluppa di correggere eventuali errori formali allo stesso modo e con gli stessi strumenti utilizzati per correggere errori sostanziali che non permetterebbero la compilazione.

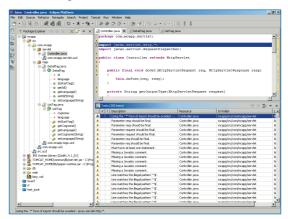



Fig. 7: Al termine dell'esecuzione del test ogni warning viene inserito nella lista dei task da eseguire di Eclipse.

CONCLUSIONI

Proviamo a rivedere gli assiomi generali da cui eravamo partiti:

- il software deve eseguire il compito per il quale è stato progettato e scritto
- il software deve essere scritto bene e deve essere manutenibile

La validità di entrambi, se il nostro obiettivo è scrivere sofware di qualità, non può essere in discussione, ma mentre per il primo ci deve essere sempre grande attenzione (pena il mancato raggiungimento degli obiettivi di progetto e spesso la perdita di qualche cliente), per il secondo si può essere, in generale, meno intransigenti in quanto un progetto software può funzionare alla perfezione anche se il codice non è scritto in maniera eccellente. In ogni caso, per motivi di manutenibilità, documentazione e comprensione dei progetti software, come sappiamo, è bene che vengano seguite in maniera precisa le procedure di qualità sintattica e di documentazione che permettano di soddisfare anche il secondo assioma. In questo articolo, oltre a descrivere la seconda parte delle regole cui dovrebbe attenersi chiunque sviluppa codice Java, è stata descritta anche una metodologia pratica di approccio al problema ed abbiamo visto come con gli strumenti giusti, e magari a costo zero, sia possibile inglobare la correzione di errori formali del codice all'interno della revisione del codice stesso, aumentando al contempo la produttività e la qualità della produzione.

Massimo Canducci

Eclipse

Eclipse è un ambiente di sviluppo Java basato su un progetto sviluppato inizialmente da IBM e donato successivamente alla comunità open source.
Il senso di Eclipse, dalla nascita del progetto, è quello di fornire agli sviluppatori degli strumenti di sviluppo sempre più sofisticati e potenti grazie ad un unico framework che integri potenzialmente tutti i linguaggi, gli Ide, i tool di modeling e gli strumenti di collaborazione esistenti.
Il framework è basato su un sistema di API pubbliche e di plug-in le cui specifiche di sviluppo sono di pubblico dominio. Questo permette a chiunque di sviluppare i propri plugin e renderli disponibili alla comunità open source oppure a pagamento.
L'ambiente completo è stato allegato al numero 71 di ioProgrammo ed è disponibile all'indirizzo www.eclipse.org.

Java e UDDI

Web Services e Java: tecniche di interrogazione.

Accedere con Java ai registri UDDI

Nei servizi Web, gioca un ruolo determinante UDDI. Acronimo di Universal Directory & Description Interface, può essere identificato da tre componenti diverse: un registro mondiale di servizi web, un insieme di specifiche ed un consorzio che li gestisce entrambi.

Nel panorama dei Web Services, Java rappresenta una componente fondamentale. Da un recente sondaggio è emerso come una grossa percentuale di sviluppatori utilizzi questo linguaggio e questa piattaforma per costruire la propria infrastruttura di servizi Web. Paradossalmente, gli sviluppatori che utilizzano Java sono di più rispetto alla loro controparte .NET, forse per via della non ancora consolidata penetrazione di mercato di quest'ultima.

LA NATURA DI UDDI

Il consorzio UDDI è un'insieme di aziende che si occupano di studiare le specifiche e di mantenere l'implementazione del registro. Tra i sostenitori del consorzio troviamo colossi come IBM, Microsoft, Ariba ed anche SUN Microsystems. Attualmente le specifiche hanno raggiunto la versione 3.0 e, come previsto dalla road-map del progetto, sono state cedute ad un ente per la standardizzazione, in particolare ad *OASIS* e *UN-CEFACT*, un ente delle nazioni unite che hanno già in gestione gli standard *ebXML*. *UDDI* dovrebbe essere l'elemento fondamentale che, insieme a *SOAP* (protocollo di comunicazione basato su XML) ed ai protocolli Internet, completa la visione dei Web Services. Questa prevede che le società espongano i propri componenti business sul Web, per essere rintracciabili da chiunque e per questo è indispensabile un meccanismo che consenta ai fornitori ed ai clienti di incontrarsi.
Questo punto di incontro dovrebbe essere costituito dai registri, una sorta di pagine gialle mondiali. In particolare, le informazioni gestite da UDDI sono:

- **pagine bianche** - informazioni anagrafiche delle aziende quali numeri telefonici, indirizzi e riferimenti Web. Sono dati che consentono un contatto a livello umano prima che tecnico;
- **pagine gialle** - catalogano i servizi e le aziende. Per via della natura mondiale del registro, le categorie hanno rilevanza globale e possono essere le più disparate. Ad esempio è possibile categorizzare un'azienda in base all'area geografica in cui opera (Europa, Italia, Milano), oppure in base al suo segmento funzionale (Business, Finanza, Prestiti, Mutui);
- **pagine verdi** - descrivono le informazioni tecniche dei servizi, per quelli integrabili direttamente a livello informatico. Un esempio sono gli URL dei servizi, o dei puntatori a file *WSDL*.

Le principali funzionalità offerte dai registri UDDI coinvolgono dunque la pubblicazione e la ricerca delle informazioni sopra descritte:

- **pubblicazione** - le funzioni di pubblicazione sul registro consentono di inviare le informazioni relative alle società ed ai servizi al registro stesso;
- **individuazione** - una volta pubblicate le informazioni, queste possono essere individuate da eventuali acquirenti tramite il processo di individuazione.

Si noti che le operazioni che coinvolgono il registro, possono essere svolte da strumenti di terze

BulkResponse

L'interfaccia *BulkResponse* permette di contenere, all'interno di un unico oggetto, un vasto insieme di oggetti eterogenei di risposta, di eccezioni e codici di stato. Come vedremo in seguito, in caso di risposta corretta, sarà possibile ottenere dalla BulkResponse una collezione contenente i dati ritornati dalla ricerca.

parti. Queste non rientrano infatti nel classico workflow relativo ai servizi Web caratterizzato primariamente dalla presenza di SOAP. Tipicamente, il protocollo di trasporto *(SOAP)*, i file di definizione (come *WSDL*) ed i registri agiscono in momenti differenti tra loro. Infatti, il protocollo di trasporto interviene quando già i programmi client e server sono stati costruiti mentre la documentazione *WSDL* interviene in fase di creazione dei client e server. Il registro interviene a monte, per identificare i *WSDL* (o informazioni equivalenti) dei servizi da interfacciare. Ma come funziona il registro globale? La cosa più interessante è che non si basa su un singolo server ma utilizza un approccio distribuito simile al *DNS*. Le informazioni vengono sempre comunicate ad uno specifico nodo, ma questo poi si occuperà di propagarle a tutti gli altri nodi *UDDI* della rete. Così, ad esempio, se comunichiamo con un nodo ospitato da IBM, le informazioni arriveranno anche agli altri nodi, ad esempio quelli ospitati da Microsoft, o da SUN. In questo modo viene garantita una certa affidabilità tramite la ridondanza dei sistemi e delle informazioni. Nonostante questo, il nodo a cui si comunicano le informazioni riveste ancora un ruolo fondamentale poiché alcune funzioni possono successive essere eseguite solo su di esso.

Fig. 1: Esistono diversi strumenti per accedere ad UDDI, il più immediato da vedere è l'interfaccia Web che propongono i costruttori, ad esempio IBM.

Quando si comunicano informazioni ad un nodo, questo diviene il *proprietario*. Funzioni successive, ad esempio l'eliminazione dei dati, possono essere eseguite solo sul nodo proprietario.

INTERFACCIARE UDDI

Le API di UDDI sono particolari: se si osservano le specifiche (vedi bibliografia) si noterà che non sono definite come specifiche di un particolare linguaggio, ma sono espresse come flussi SOAP. Le primitive di UDDI sono infatti esse stesse servizi Web. In questo modo non é stato necessario definire un insieme di API per ciascuna piattaforma supportata: la piattaforma nativa di UDDI é quella dei servizi Web. Nel mondo Java, per operare con UDDI esistono dunque tre differenti strategie:

- **SOAP**. Si possono utilizzare *SAAJ* e *JAXM* per dialogare direttamente in SOAP con il registro. È inoltre possibile pensare di recuperare i *WSDL* dei servizi UDDI e di utilizzare *JAX-RPC* per costruire un modello Java delle chiamate UDDI.
- **API proprietarie**. Fornitori di software di base per i servizi Web offrono prodotti di integrazione che oltre al supporto a SOAP e WSDL forniscono API molto semplici per l'accesso ad UDDI.
- **JAXR (Java API for XML Registries)**. E' lo standard della piattaforma Java per l'accesso ai registri di servizi Web. L'implementazione di riferimento é contenuta nel *JWSDP* (*Java Web Services Developer Pack*), disponibile per il download al sito ufficiale di SUN.

Operare direttamente con SOAP può non essere facile: è necessario trattare dati in XML e gestire le comunicazioni più o meno a basso livello. La scelta di operare con API proprietarie o standard è sempre delicata: se da una parte le API proprietarie possono essere più semplici, dall'altra costringono all'utilizzo del software di un particolare produttore.
Ad ogni modo, le API standard garantiscono una certa scelta per i fornitori di software anche se possono risultare più complesse.

JAXR

Queste API di SUN risultano abbastanza complesse, non solo perché non sono legate ad uno specifico prodotto (in questo caso non avrebbe fatto molta differenza) ma più per il fatto che godono di un'architettura multi-registro. Le API *JAXR* non sono infatti vincolate solo all'utilizzo di UDDI ma possono interagire anche con altri tipi di registro come *ebXML-R*. Questo è l'altro standard mondiale per la creazione dei registri Web, meno famoso in quanto per certi versi meno "considerato" di UDDI. Con il fatto che le specifiche UDDI sono ora in gestione dallo stesso consorzio che ha in mano *ebXML-R,* è possibile pensare ad una convergenza delle specifiche e quindi ad una minore importanza dell'architettura così aperta di *JAXR*. La cosa da segnalare è che il modello informativo (le classi che definiscono le entità presenti nel registro) di *JAXR* é in realtà mutuato da *ebXML*. Inoltre, le *API JAXR* dispongono, oltre che delle chiamate generalizzate per operare con la pubblicazione e la ricerca delle informazioni su generici registri, anche di API specifiche per UD-

Java e UDDI

Incontri

UDDI dovrebbe essere l'elemento fondamentale che, insieme al protocollo di comunicazione basato su XML ed ai protocolli Internet, completa la visione dei Web Services. Questa prevede che le società espongano i propri componenti di business sul Web per essere rintracciabili da chiunque e per questo è indispensabile un meccanismo che consenta ai fornitori ed i clienti di incontrarsi reciprocamente.

Java e UDDI

DI, che hanno l'obiettivo di offrire API familiari e semplici da utilizzare allo sviluppatore abituato a questo registro. Senza addentrarci nei complessi meandri di *JAXR*, verrà analizzato un programma di esempio per l'accesso al registro UDDI che esegue la ricerca di una azienda. Il programma é presente nel JWSDP nel file *BusinessQueryTest.java*.

CONNESSIONI E MANAGER

Come molte altre API della piattaforma Java, JAXR fa largo uso di design pattern, in particolare del pattern *futures* e di *factory*. Quest'ultimo viene utilizzato per gestire le connessioni al registro: tutte le operazioni possibili in *JAXR* sono effettuabili agendo su oggetti in qualche modo forniti da una *connessione al registro*. Le connessioni vengono fornite al programma tramite la classe *ConnectionFactory*. Questa dispone del metodo *newInstance()* che resituisce una *factory* che può essere utilizzata per stabilire le connessioni. Sono molti i parametri che possono intervenire in fase di connessione ad un registro ed alcuni potrebbero essere specifici in funzione di un dato registro. Questi parametri vanno passati alla *factory* tramite il metodo *setProperties()* sotto forma di oggetto *Properties*. I basilari sono:

- **javax.xml.registry.queryManagerURL** - Indica l'URL di accesso al registro per le operazioni di ricerca.
- **javax.xml.registry.lifeCycleManagerURL** - Indica l'URL di accesso al registro per le operazioni di pubblicazione.
- **javax.xml.registry.factoryClass** - Indica la classe *JAXR* che implementa la *factory*. Ad esempio:

```
com.sun .xml.registry.uddi.ConnectionFactoryImpl
```

- **com.sun.xml.registry.http.proxyHost** - Indica il nome del computer che funge da proxy per l'accesso ad Internet.
- **com.sun.xml.registry.http.proxyPort** - Indica la porta del computer che funge da proxy per l'accesso ad Internet.
  Una volta ottenuta la connessione é possibile passare alla richiesta della connessione, tramite il metodo *createConnection():*

```
Connection conn = factory.createConnection();
```

A questo punto è possibile ottenere l'oggetto che si occupa di eseguire la query in modalità semplificata: il *BusinessQueryManager*. Per prima cosa é necessario passare dall'oggetto che rappresenta il servizio del registro per poi passare al vero e proprio *BusinessQueryManager*:

```
RegistryService rs = conn.getRegistryService();
BusinessQueryManager bqm =
                    rs.getBusinessQueryManager();
```

Tramite l'oggetto *BusinessQueryManager* é ora possibile cercare le aziende che rispondono a determinati criteri, tramite il metodo *findOrganizations()*.

CERCARE CON JAXR

A questo punto è bene introdurre alcuni concetti peculiari di *JAXR*. Per prima cosa, è interessante notare come è conformato il codice che esegue la ricerca: essendo i registri Web strutturati per contenere una miriade di informazioni, risulta di fondamentale importanza la possibilità di applicare ricerche "complesse". JAXR viene in contro a questa necessità consentendo di specificare una serie di parametri di selezione. Ad esempio, per la ricerca di aziende é possibile specificare:

- **findQualifiers -** Una serie di qualificatori definiti in accordo all'interfaccia *FindQualifier* che permettono di definire i criteri di base utilizzati per l'ordinamento, il confronto ed altri criteri similari per le stringhe.
- **namePatterns -** Contiene un elenco di pattern tipo SQL con i nomi delle aziende da cercare. Ad esempio *"SU%"* cerca tutte le aziende che cominciano con *"SU"*. Se vengono specificati più pattern, la ricerca é cumulativa, a meno che i *findQualifiers* specifichino un comportamento differente.
- **classifications -** Individuano le classificazioni a cui i nominativi devono appartenere. Ad esempio, se il nome di una società risponde ai precedenti parametri ma é catalogata come *"Francese"* e le classificazioni richieste specificano solo *"Italiana"*, questa non verrà ritornata nella ricerca.
- **specifications -** Individua le specifiche tecniche dei servizi Web; il funzionamento é analogo alle classificazioni: vengono ritornate solo le aziende in grado di fornire determinati servizi identificati dalle *specifications*.
- **externalIdentifiers -** Opera in modo similare alle precedenti ma tratta di identificativi esterni, come le classificazioni *DUNS*.
- **externalLinks -** Restringe la ricerca alle entità che supportano i link esterni specificati. I link

Diritti

Quando si comunicano informazioni ad un nodo, questo diviene il proprietario. Funzioni successive, ad esempio l'eliminazione dei dati, possono essere eseguiti solo sul nodo proprietario.

Java e UDDI

esterni puntano a dettagli tecnici necessari per invocare il servizio.

Se non si desidera utilizzare uno o più parametri di ricerca, è sufficiente passare *null* come parametro. La chiamata *findOrganizations()* ritorna come risposta un oggetto *BulkResponse*. Questo ci consente di affrontare due concetti di *JAXR:* per prima cosa, si può notare che tra le strutture dati e le API esiste un accoppiamento lasco. Questo è dovuto al fatto che i registri sono basati su servizi SOAP. Un servizio SOAP può ritornare strutture dati eterogenee, in funzione della tipologia di risposta da dare: un approccio sicuramente differente rispetto al linguaggi *strong-typed* come Java che richiedono, ad esempio, l'indicazione specifica di un tipo per il valore di ritorno di un metodo. L'interfaccia *BulkResponse* permette di contenere, all'interno di un unico oggetto, un vasto insieme di oggetti eterogenei di risposta, di eccezioni e codici di stato. Come vedremo in seguito, in caso di risposta corretta, sarà possibile ottenere dalla *BulkResponse* una collezione contenente i dati ritornati dalla ricerca. Un'altra particolarità si nasconde dietro gli oggetti ritornati in *BulkResponse* (e che fanno parte del modello informativo di *JAXR*): questi adottano il già citato pattern *futures*, detto anche *lazy loading*. Questo particolare pattern, a cui ci si riferisce anche come *loading on-demand*, prevede che gli oggetti del modello informativo vengano subito ritornati al client, senza che il caricamento di tutte le informazioni sia completato. Solo al momento dell'invocazione di un particolare metodo *getter*, l'informazione richiesta verrà recuperata (eventualmente insieme ad altri dati) allo scopo di ottimizzare le comunicazioni di rete. Questo approccio ha una serie di conseguenze: quella più immediata è la maggiore responsività di *JAXR* che, dovendo estrarre un numero limitato di informazioni (tipicamente le chiavi), ha un minore carico sulla rete e sul processore. Una seconda conseguenza è però la necessità di mantenere il computer collegato alla rete: contrariamente, un accesso ad informazioni non presenti nella memoria locale genererà un errore di comunicazione.

ESTRARRE LE INFORMAZIONI

L'oggetto *BulkResponse*, oltre alle informazioni vere e proprie, contiene metadati che indicano lo stato della richiesta. Ad esempio, attraverso la chiamata *getStatus()* è possibile verificare l'esito della richiesta:

```
if (br.getStatus() == JAXRResponse.STATUS_SUCCESS)
```

L'interfaccia *JAXResponse* definisce, oltre agli stati delle risposte anche una superinterfaccia per tutte le risposte *JAXR*, *BulkResponse* inclusa. Per estrarre le informazioni dalla *BulkResponse* é necessario utilizzare il metodo *getCollection()*. Il metodo ritorna una collezione Java che conterrà oggetti del tipo richiesto, nel caso specifico, oggetti di tipo *Organization*.

```
Collection orgs = br.getCollection();
```

A questo punto l'estrazione delle informazioni si traduce in un esercizio di utilizzo degli iteratori di Java2:

```
Iterator iter = orgs.iterator();
while (iter.hasNext()) {
Organization org = (Organization) iter.next();
//...
}
```

Una volta in possesso di un oggetto *Organization*, é possibile interrogarlo per ottenerne i dettagli informativi, come ad esempio l'elenco dei servizi offerti. Questo é un esempio dell'utilità del *lazy-loading:* il caricamento di tutte le informazioni di una organizzazione, compresi tutti i servizi e tutti i dettagli di tutti i servizi sarebbe stata un'operazione molto onerosa, senza questo meccanismo. In questo modo le prestazioni rimangono accettabilissime. L'ottenimento dei servizi avviene in modo similare a quanto visto per le organizzazioni:

```
Collection services = org.getServices();
Iterator siter = services.iterator();
while (siter.hasNext()) {
Service service = (Service) siter.next();
//...
}
```

In caso di esito negativo, i problemi potrebbero essere di diversa natura. È per questo che la risposta *BulkResponse* potrebbe contenere una collezione di eccezioni. In questo caso é necessario estrapolarle singolarmente allo stesso modo utilizzato per le organizzazioni ed i servizi.

CONCLUSIONI

Sebbene le tecnologie per i registri di servizi stiano guadagnando terreno, non sono ancora molto diffuse. SUN e SAP hanno recentemente dichiarato ad una conferenza per sviluppatori come in realtà i loro clienti stiano sì considerando l'utilizzo di UDDI, ma più per la creazione di registri interni invece che per l'interfacciamento servizi di terze parti su Internet. Forse il fatto che *l'80%* degli URL presenti nel registro UDDI mondiale è sbagliato *(rif. TechDays 2001)* è uno dei motivi.

Massimiliano Bigatti

Bibliografia

http://www.uddi.org/specification.html

Sul Web

[ws1] http://www.webservices.org

[uddi] http://www.uddi.org

[ebxml] http://www.ebxml.org

[ibm1] http://www-3.ibm.com/services/uddi/

[sun] http://java.sun.com/webservices

INBox L'esperto risponde...

Power Users

Buongiorno Ing. Florio, ogni tanto mi trovo "costretto" a chiedere qualche consiglio, ma proprio quando non so più che strada prendere (salto i meritati complimenti alla Sua persona e alla rivista); sapevo che con Windows 2000 Pro il gruppo Power Users permetteva l'installazione di software sul sistema; oggi ho creato un utente appartenente a questo gruppo e ho provato ad installare del software.
Risultato: impossibile installare perché richiedono i privilegi di amministratore!
C'è qualche maniera per abilitare il gruppo Power User ad installare software senza che manovre incaute possano danneggiare il sistema?

Risponde Elia Florio.

Ringrazio sentitamente per i complimenti che continuamente ricevo da voi lettori.
Si, in genere questo è vero, gli utenti *"Power Users"* possono installare determinati software sul sistema operativo. Purtroppo ci sono installazioni ed installazioni... Power Users dà ad un utente il permesso di installare i programmi, tuttavia se il software installato non si limita a fare le operazioni "legali", ma prova ad esempio a scrivere nella WINDOWS\SYSTEM o a modificare DLL di sistema e cose di questo tipo, scattano le restrizioni.
Questa limitazione è necessaria, perché altrimenti Power Users potrebbe installare Trojan, Worm o potrebbe eseguire quei programmi che loggano le password di sistema o che trasformano un utente generico in Administrator.
Per abilitare il gruppo Power Users, senza problemi di manovre incaute, è necessario avere un dominio Windows 2000 (con relativo server) sul quale impostare restrizioni e permessi a gruppi di utenti.

Corso Java... delucidazioni

Gentile redazione, sono un nuovo programmatore a cui piace la rivista ma ho alcuni problemi con il corso di Java. Dico subito che ho esperienze con il C++ e questo mi ha permesso di capire molte cose, in quanto i due linguaggi di programmazione si somigliano molto. Il problema è sorto quando ho installato J2SE SDK ed ho provato a compilare ed eseguire la versione Java del programma *Ciao Mondo*.
Lo *javac* funziona benissimo, il problema sorge quando cerco di eseguire il programma con la VM *java.exe*. Eseguendo il file *Saluti.class* il Prompt compare per un microsecondo e non ho il tempo di leggere se il programma è eseguito con successo. Può essere un problema dato dalla compatibilità con il kit Borland per C++ versione 3.1 o il problema è un'altro? Vi sarei grato se mi segnalereste la VM per eseguire codice Java in ambiente Linux Mandrake o comunque in Linux.
Grazie e complimenti per la rivista.

Matteo Uberti • Email

Risponde Paolo Perrotta

Ciao Matteo, la soluzione del tuo problema (se ho capito qual è il problema) è semplice: anziché eseguire il file con il comando *"Esegui"* del menu di Windows, apri una finestra del prompt permanente. Il modo più semplice per farlo è andare nel menu *"Start"*, scegliere *"Esegui"* e inserire il comando *"cmd"*. Si dovrebbe aprire una finestra del prompt, nella quale potrai comodamente lanciare il programma ed esaminarne l'output. Se il programma viene eseguito con successo, dovresti leggerne l'output subito prima del prompt successivo. Per quanto riguarda la VM Java per Linux, la trovi all'indirizzo *http://java .sun.com*. Puoi scaricare l'ambiente di run-time o l'intero SDK per Linux.

Spiare in Rete

Buon giorno a tutti voi, innanzi tutto vorrei complimentarmi per la qualità della vs rivista, che compro ogni mese da almeno 3 anni, sempre chiarissima e completa nei vari argomenti trattati. Vorrei portare alla vs attenzione un problema che non riesco a risolvere a proposito dell'articolo "Spiare in rete" del n° Luglio/Agosto 2002 di Elia Florio. Usando il Borland C++ Builder anziché il VB per compilare l'eseguibile che accede alla DLL che contiene la procedura di Hook della tastiera, ritorna l'errore di *"Entry point not found"*, quindi non riesce a trovare la funzione *KeyProc.*
Devo specificare che ho fatto una semplice "traduzione" da VB a C++ utilizzando le stesse API usate nel programma VB e che la DLL l'ho generata con il Builder stesso.
Il problema può nascere dal fatto che ho compilato la DLL con il Builder, o invece da qualche arrangiamento mancante nell'eseguibile vero e proprio?Se dovessi implementare un Hook per il Mouse,cosa dovrei cambiare?

Avete dei suggerimenti da darmi? Vi ringrazio in anticipo per la risposta che vorrete inviarmi.

Mauro Casula

Risponde Elia Florio

Si tratta di un errore tipico...in pratica l'eseguibile che crei con BCC non riesce a trovare l'entry point della libreria DLL, cioè non riesce a localizzare il punto d'ingresso per una delle chiamate fatte alla DLL. Deve esserci qualche problema nella "traduzione", magari un errore di ortografia nei nomi delle API, oppure una sciocchezza simile. Quando esegui il file EXE sei certo che la DLL si trovi nella stessa directory? Il codice per l'intercettazione del mouse è un po' diverso, nel senso che il modello di hook globale è molto simile, cambiano le API che accedono al driver della periferica.

Script Control nelle applicazioni

Amici di ioProgrammo, vi scrivo per avere delucidazioni in merito allo sviluppo di un'applicazione che mi permetta di utilizzare i famosi linguaggi di scripting JavaScript e VBScript all'interno delle mie applicazioni Delphi. Sicuro di un vostro "pronto intervento" vi ringrazio e vi auguro buona continuazione di lavoro.

Mauro • Email

Gentile lettore, Microsoft ha ideato un insieme di interfacce COM, denominate ActiveScripting, per comunicare direttamente con un motore di scripting.

Solitamente solo gli "addetti ai lavori" più esigenti avvertono la necessità di scontrarsi con l'astrusità di *ActiveScripting*. Per implementare un motore di scripting basta importare, all'interno della nostra applicazione, l'oggetto *Microsoft Script Control* cliccando sulla voce *Import ActiveX Control* del menu *Component* di Delphi. Sarà così creata la unit *MSScriptControl_TLB* nella cartella predefinita e aggiunto un componente alla palette.

Un banale esempio di utilizzo del componente *TScriptControl* richiede i seguenti passi.

Supponiamo di avere un main form con all'interno un bottone denominato *BitBtnl* e un controllo testo denominato *MemoScript*, quest'ultimo servirà all'utente per digitare lo script da mandare in esecuzione.

1) Posizioniamo il componente *TScriptControl* sul form

2) Nella procedura *FormCreate* stabiliamo il linguaggio dello script ed eventuali altre proprietà:

```
procedure TForm1.FormCreate(
                                    Sender: TObject);
begin
// hWnd del controllo "genitore" =
                                         Form1.Handle
 ScriptControl1.SitehWnd := Self.Handle;
// Linguaggio da utilizzare = VBScript
 ScriptControl1.Language := 'VBScript';
end;
```

3) Associamo al pulsante *BitBtnl* il seguente codice:

```
procedure TForm1.BitBtn1Click(
                                    Sender: TObject);
begin
 ScriptControl1.ExecuteStatement(
                                   MemoScript.Text);
end;
```

Con pochissime righe di codice abbiamo "creato" un interprete di codice VBScript/JScript , in grado di eseguire qualsiasi istruzione.
L'esecuzione di interi script è un compito meno elementare perché il nome della funzione ed i relativi parametri devono essere opportunamente passati a *ScriptControl*.

Cos'è SQLXML?

Una domanda alla quale spero possiate dare risposta al più presto: di recente ho sentito parlare, in merito a SQL Server 200, di SQLXML; la cosa mi intriga molto, essendo uno sviluppatore che da poco si sta affacciando all'affascinante mondo di XML.

Carlo Somino • Email

L'utilizzo di XML come standard per l'interscambio delle informazioni ha richiesto, e richiede ancora oggi, degli sforzi per creare documenti XML basati su dati contenuti in tabelle di database. Lo sforzo solitamente consiste nella creazione di un traduttore che a partire dalle tabelle interessate, crei i documenti XML da usare come file di interscambio. Di questa necessità Microsoft ha fatto virtù, creando un modello di accesso ai dati completamente basato su XML e completamente integrato in SQL 2000, ovvero SQLXML. Le possibilità offerte dal modello di accesso ai dati integrato SQLXML sono molteplici:

- Accesso al motore di query di SQL 2000 attraverso il protocollo http, mediante query "*in-line*" o mediante l'uso di appositi "*Templates*";
- Estrazione diretta dei dati in formato XML nativo;
- Estrazione e trasformazione dei dati XML mediante XSLT *(eXtensible Stylesheet Language Transformations);*
- Modifica ed inserimento dati mediante *UpdateGrams;*
- Potente motore di *Bulk-Load* di dati; il *Bulk-Load* offre prestazioni superiori quando risulta necessario caricare grandi quantità di dati all'interno del nostro server di database;
- Supporto, tramite le classi *managed*, del Framework .NET;
- Supporto dei Web Service grazie al SOAP Toolkit 2.0;
- Formattazione dei risultati XML in modalità *Client-Side* o *Server-Side.*

Per contattarci:
e-mail: iopinbox@edmaster.it
Posta: Edizioni Master,
Via Cesare Correnti, 1 - 20123 Milano

Biblioteca

ON LINE

Game Development Search Engine

Il punto di incontro per tutti i programmatori di videogame, siano essi aspiranti tali o provetti sviluppatori. Diverse le aree di interessate, tra queste citiamo: 3D e grafica, articoli e tutorial, codice sorgente e newsgroup.

http://www.gdse.com/servlet/gdse.main

Java World

Semplicemente uno dei più vasti container su Java. Articoli, faq, libri, tip e quanto altro possa essere di supporto allo sviluppo di applicazioni.

http://www.javaworld.com

The Code Project

Una marea di articoli per ogni linguaggio di programmazione. Un sito creato dagli sviluppatori per gli sviluppatori.

www.codeproject.com

Comprehensive VB.NET Debugging

Lo sviluppo di applicazioni, è sempre più difficile, è sovente incorrere in errori di progettazione che, in alcuni casi, possono rivelarsi deleteri.
Il testo in oggetto pone l'accento sui tool di Microsoft .NET, messi a disposizione dello sviluppatore, per cercare di eliminare bug che, spesso e volentieri, rimangono "invisibili" al programmatore.
In particolar modo si fa riferimento a quanto di buono gli sviluppatori di Redmond hanno implementato in Visual Basic .NET. Oltre ai tool, l'autore del testo in oggetto analizza anche i classici problemi di programmazione, mostrando, in modo molto dettagliato, tutte le fasi di debug che portano a "scoprire" le vulnerabilità del codice.

Difficoltà: Medio – Alta • Autore: M.Pearce • Editore: Apress http://www.apress.com
ISBN: 1-59059-050-3 • Anno di pubblicazione: 2003 • Lingua: Inglese • Pagine: 528
Prezzo: $ 54.99

MySQL 4 – Guida Completa

MySQL è un RDBMS Open Source, quindi "libero", a detta di molti è il database più veloce e funzionale rispetto anche a database di categorie superiori. Il testo si prefigge di introdurre agli utenti questo potente sistema e di condurli nella realizzazione e nella gestione di un database di grandi dimensioni e prestazioni. Il libro tratta della versione 4.0.1, che gira sia sotto Linux sia sotto Windows. Destinato a chi sta cercando di sostituire il vecchio DB con strumenti più efficaci, a sviluppatori di siti web che si interfacceranno con i database, a chi usa la piattaforma Linux e ha bisogno di un sistema di archiviazione dati di livello aziendale e anche a chi vuole, comunque, imparare a usare i database relazionali.

Difficoltà: Media • Autore: A.Butcher • Editore: Apogeo http://www.apogeonline.com
ISBN: 88-503-2127-9 • Anno di pubblicazione: 2003 • Lingua: Italiano • Pagine: 544 • Prezzo: € 39,00

Hack Attacks Testing

Un testo che può rivelarsi fondamentale per ogni amministratore di sistema che, ogni giorno, si trova dinanzi al problema di garantire la sicurezza della propria Rete. Scritto da un guru della sicurezza informatica, il testo mostra gran parte delle tecniche di hackeraggio che possono compromettere la vulnerabilità del sistema, altresì viene mostrato come difendersi da tali attacchi, analizzando passo passo diverse strategie. Da segnalare:

- Installazione e configurazione "sicura" di Windows 2000 Server, Linux, Solaris, Mac OS X.
- ISS, CyberCop, Nessus, SAINT, e STAT scanner.
- Diversi tool di analisi per testare la sicurezza del proprio sistema operativo.

Il CD-Rom allegato al testo contiene una simulazione di scanner alla ricerca di possibili vulnerabilità, una versione evaluation di ISS Internet Scanner e diverse altre utilità.

Difficoltà: Media • Autore: J.Chirillo • Editore: Wiley http://www.wileyeurope.com
ISBN: 0-471-22946-6 • Anno di pubblicazione: 2003 • Lingua: Italiano • Pagine: 560
Prezzo: € 47.70 • Contiene 1 CD-Rom